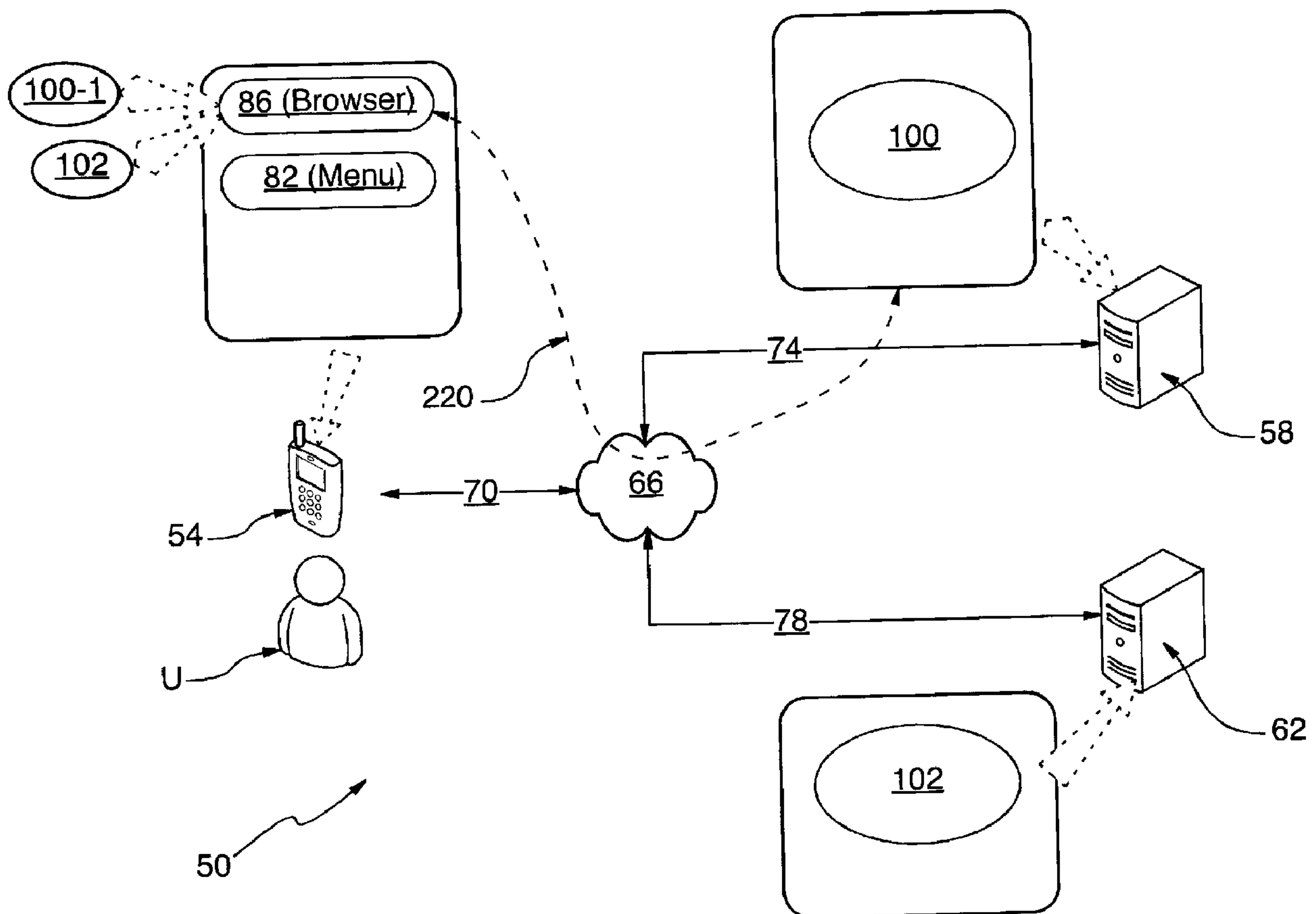




(86) **Date de dépôt PCT/PCT Filing Date:** 2008/05/12
(87) **Date publication PCT/PCT Publication Date:** 2008/11/27
(45) **Date de délivrance/Issue Date:** 2014/11/18
(85) **Entrée phase nationale/National Entry:** 2009/11/17
(86) **N° demande PCT/PCT Application No.:** CA 2008/000903
(87) **N° publication PCT/PCT Publication No.:** 2008/141424
(30) **Priorité/Priority:** 2007/05/17 (US60/924,503)

(51) **Cl.Int./Int.Cl. H04L 12/16** (2006.01),
G06F 17/00 (2006.01), **G06F 3/0482** (2013.01)
(72) **Inventeurs/Inventors:**
MALICK, MARTYN HENRI, CA;
KIM, SANG-HEUN, CA;
STINSON, CHARLES LAURENCE, CA
(73) **Propriétaire/Owner:**
BLACKBERRY LIMITED, CA
(74) **Agent:** PERRY + CURRIER

(54) **Titre : SYSTEME ET PROCEDE DE NAVIGATION DE CONTENU**
(54) **Title: SYSTEM AND METHOD FOR CONTENT NAVIGATION**



(57) **Abrégé/Abstract:**

The present specification provides, amongst other things, a method and system for navigating content. In an embodiment a portable electronic device is provided having a browser application and a native menu application. The embodiment also includes a



(57) Abrégé(suite)/Abstract(continued):

network that interconnects a web-server and said portable electronic device. The web-server hosts web pages that include menus and content. The portable electronic device is configured to obtain a schema respective to the web-pages whereby the web-page menus can be generated on the portable electronic device using the native menu application rather than the browser application, thereby permitting navigation of content on the portable electronic device via the native menu application.

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
27 November 2008 (27.11.2008)

PCT

(10) International Publication Number
WO 2008/141424 A1

(51) International Patent Classification:

H04L 12/16 (2006.01) *H04Q 7/22* (2006.01)
G06F 17/00 (2006.01) *G06Q 30/00* (2006.01)
G06F 3/048 (2006.01)

(21) International Application Number:

PCT/CA2008/000903

(22) International Filing Date: 12 May 2008 (12.05.2008)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

60/924,503 17 May 2007 (17.05.2007) US

(71) Applicants (for all designated States except US): **RESEARCH IN MOTION LIMITED** [CA/CA]; 295 Phillip Street, Waterloo, Ontario N2L 3W8 (CA). **FAT FREE MOBILE INC.** [CA/CA]; 3872 Swiftdale Drive, Mississauga, Ontario L5M 6M2 (CA).

(72) Inventors; and

(75) Inventors/Applicants (for US only): **MALLICK, Martyn** [CA/CA]; 2601 Erb's Road, Baden, Ontario N3A 3M4 (CA). **KIM, Sang-Heun** [CA/CA]; 2610-33 Elm Drive West, Mississauga, Ontario L5B 4M2 (CA). **STINSON, Charles, Laurence** [CA/CA]; 3872 Siftsdale Drive, Mississauga, Ontario L5M 6M2 (CA).

(74) Agent: **CURRIER, Andrew, T.**; PERRY + CURRIER INC., 1300 Yonge Street, Suite 500, Toronto, Ontario M4T 1X3 (CA).

(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declaration under Rule 4.17:

— as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))

[Continued on next page]

(54) Title: SYSTEM AND METHOD FOR CONTENT NAVIGATION

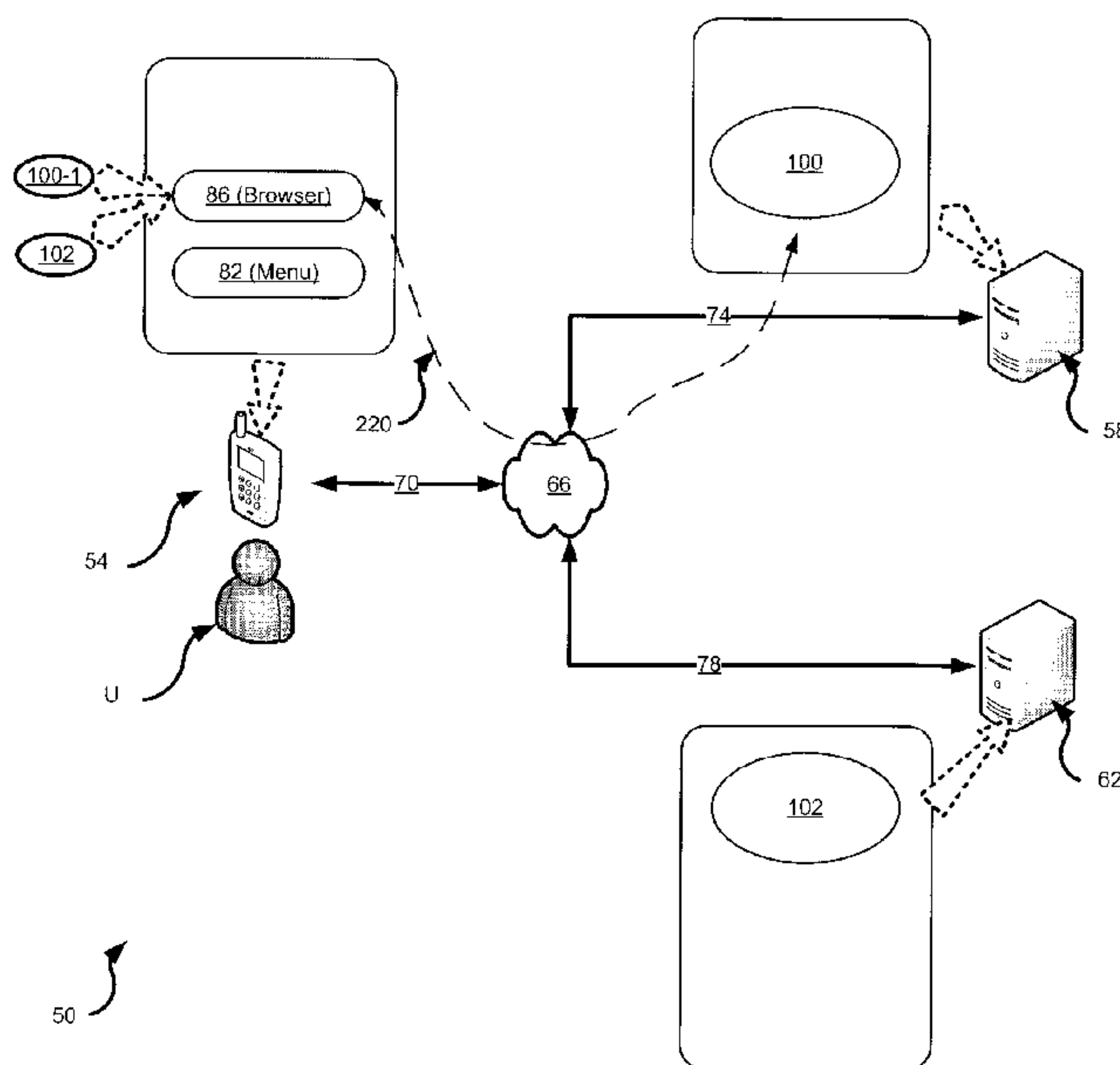


Fig. 11

(57) Abstract: The present specification provides, amongst other things, a method and system for navigating content. In an embodiment a portable electronic device is provided having a browser application and a native menu application. The embodiment also includes a network that interconnects a web-server and said portable electronic device. The web-server hosts web pages that include menus and content. The portable electronic device is configured to obtain a schema respective to the web-pages whereby the web-page menus can be generated on the portable electronic device using the native menu application rather than the browser application, thereby permitting navigation of content on the portable electronic device via the native menu application.

WO 2008/141424 A1

WO 2008/141424 A1



Published:

— *with international search report*

— *with information concerning request for restoration of the right of priority in respect of one or more priority claims*

SYSTEM AND METHOD FOR CONTENT NAVIGATION

FIELD

[0001] The present specification relates generally to telecommunication and more specifically relates to a system and method for content navigation.

BACKGROUND

[0002] Computing devices are becoming smaller and increasingly utilize wireless connectivity. Examples of such computing devices include portable computing devices that include wireless network browsing capability as well as telephony and personal information management capabilities. The smaller size of such client devices necessarily limits their display capabilities. Furthermore the wireless connections to such devices typically have less bandwidth than corresponding wired connections. The Wireless Application Protocol ("WAP") was designed to address such issues, but WAP can still provide a very unsatisfactory experience or even completely ineffective experience, particularly where the small client device needs to effect a connection with web-sites that host web-pages that are optimized for full traditional desktop browsers.

[0003] Document US-2002/054161-A1 discloses a method, system, and computer-readable medium for allowing easier access to the Web pages of a Web site is provided in which a requested Web page that includes a menu is downloaded along with a first executable component comprising display instructions assigned the Web site. A second executable component is also downloaded with the Web page for creating a data structure in memory having a tree-like structure and including the titles of the Web site's Web pages and links to the titled Web pages. A portion of a cascading menu is generated based on the executable component and the data structure when a corresponding portion of the menu is selected. The executable components are automatically updated, if it is determined that more current versions exist.

[0004] Document WO-2004/109557-A1 discloses a method of displaying and navigating information using a hierarchical, spring loaded DHTML menu system that adjusts to fit the boundaries of a resizable, container by using a set of standard HTML table row elements having a hierarchical dependency that is consistently displayed as a web-page, under all reasonable resizing and is done using a strut row that produces the largest width requirement when the table structure is resized to occupy a small width.

[0005] Document WO-01/86462-A1 discloses a method for converting HTML/XML to HDML/WML in real time for display on wireless communication devices. The request for Web content comes from the mobile unit and is received by a proxy server. The proxy server retrieves the content from the Web or from local cache. The proxy server parses the mobile unit request to determine the mobile device type and the mobile user preferences. The specifications for the device type and user preferences are retrieved by the proxy server from its database. Based on the device and user information, the proxy server retrieves a conversion specification tailored to the device and the user. The retrieved Web content is then processed by the proxy server based on the conversion specification. The result of this processing is then optimized for transmission and sent to the waiting wireless communications devices.

SUMMARY

[0006] The present specification provides, amongst other things, a method and system for navigating content. In an embodiment a portable electronic device is provided having a browser application and a native menu application. The embodiment also includes a network that interconnects a web-server and said portable electronic device. The web-server hosts web pages that include menus and content. The portable electronic device is configured to obtain a schema respective to the web-pages whereby the web-page menus can be generated on the portable electronic device using the native menu application rather than the browser application, thereby permitting navigation of content on the portable electronic device via the native menu application.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] Figure 1 is schematic representation of a system for content navigation.

[0008] Figure 2 is a schematic representation of a wireless communication device from Figure 1.

[0009] Figure 3 is a schematic representation of a display and a portion of a keyboard of the device of Figure 1, wherein the display is showing a screen from a contact manager application.

[0010] Figure 4 is a schematic representation of the display and the portion of a keyboard of Figure 3, wherein the display is showing the screen from contact manager application

including a menu from a menu application with menu selections that are contextual to the contact manager application.

[0011] Figure 5 shows an exemplary web-page available from the web-server in Figure 1.

[0012] Figure 6 shows an exemplary web-page available from the web-server in Figure 1.

[0013] Figure 7 shows an exemplary web-page available from the web-server in Figure 1.

[0014] Figure 8 shows an exemplary web-page available from the web-server in Figure 1.

[0015] Figure 9 shows a flowchart depicting a method for content navigation.

[0016] Figure 10 shows the system of Figure 1 during exemplary performance of part of the method of Figure 9.

[0017] Figure 11 shows the system of Figure 1 during exemplary performance of part of the method of Figure 9.

[0018] Figure 12 shows the display of the client machine of Figure 1 during exemplary performance of part of the method of Figure 9.

[0019] Figure 13 shows the display of the client machine of Figure 1 during exemplary performance of part of the method of Figure 9.

[0020] Figure 14 shows the display of the client machine of Figure 1 during exemplary performance of part of the method of Figure 9.

[0021] Figure 15 shows the display of the client machine of Figure 1 during exemplary performance of part of the method of Figure 9.

[0022] Figure 16 shows the display of the client machine of Figure 1 during exemplary performance of part of the method of Figure 9.

[0023] Figure 17 a schematic representation of a system for content navigation in accordance with another embodiment.

[0024] Figure 18 shows a flowchart depicting a method for content navigation in accordance with another embodiment.

[0025] Figure 19 shows the display of the client machine of Figure 17 during exemplary performance of part of the method of Figure 18.

[0026] Figure 20 is schematic representation of a system for content navigation.

[0027] Figure 21 is a schematic representation of a wireless communication device from Figure 20.

[0028] Figure 22 shows the process of converting webpages to a hierarchical structure.

[0029] Figure 23 shows a flowchart depicting a method to obtain and satisfy a web page request.

[0030] Figure 24 shows the system of Figure 20 during exemplary performance of part of the method of Figure 23.

[0031] Figure 25 shows exemplary perspectives of browsing sessions.

[0032] Figure 26 shows an exemplary web-page available from the web-server in Figure 20.

[0033] Figure 27 shows exemplary output from the Schema Engine.

[0034] Figure 28 shows an exemplary rendering of a mobile application.

[0035] Figure 29 shows an exemplary menu rendered on the mobile device.

[0036] Figure 30 shows an exemplary web-page available from the web-server in Figure 20.

[0037] Figure 31 shows an exemplary web-page available from the web-server in Figure 20.

[0038] Figure 32 shows an exemplary web-page available from the web-server in Figure 20.

[0039] Figure 33 shows an exemplary web-page available from the web-server in Figure 20.

[0040] Figure 34 shows an exemplary web-page available from the web-server in Figure 20.

[0041] Figure 35 shows action 1 of Figure 44.

[0042] Figure 36 shows action 3 of Figure 44.

[0043] Figure 37 shows action 5 of Figure 44.

[0044] Figure 38 shows action 7 of Figure 44.

[0045] Figure 39 shows action 9 of Figure 44.

[0046] Figure 40 shows an exemplary web-page available from the web-server in Figure 20.

[0047] Figure 41 shows an exemplary result of the process of assisted capturing of the web-page in Figure 40.

[0048] Figure 42 shows an exemplary rich bookmarks list.

[0049] Figure 43 shows an exemplary rich bookmarks list.

[0050] Figure 44 shows an exemplary action of browsing through ABC ComTech Corp. to purchase an item.

[0051] Figure 45 shows an exemplary request/response for a page the website in Figure 20.

DETAILED DESCRIPTION OF THE EMBODIMENTS

[0052] Referring now to Figure 1, a system for content navigation in a computing device is indicated generally at 50. In a present embodiment system 50 comprise a first computing device in the form of a client machine 54 and at least one additional computing device implanted as a second computing device in the form of a web-server 58 and a third computing device in the form of a schema server 62. A network 66 interconnects each of the foregoing components.

[0053] Each client machine 54 is typically any type of computing or electronic device that can be used to interact with content available on network 66. Each client machine 54 is operated by a user U. Interaction includes displaying of information on client machine 54 as well as to receive input at client machine 54 that is in turn sent back over network 66. In a present embodiment, client machine 54 is a mobile electronic device with the combined functionality of a personal digital assistant, cell phone, email paging device, and a web-browser. Such a mobile electronic device thus includes a keyboard (or other input device(s)), a display, a speaker, (or other output device(s)) and a chassis within which the keyboard, display monitor, speaker are housed. The chassis also houses one or more central processing units, volatile memory (e.g. random access memory), persistent memory (e.g. Flash read only memory) and

network interfaces to allow machine 54 to communicate over network 66.

[0054] Referring now to Figure 2, a schematic block diagram shows client machine 54 in greater detail. It should be emphasized that the structure in Figure 2 is purely exemplary, and contemplates a device that be used for both wireless voice (e.g. telephony) and wireless data (e.g. email, web browsing, text) communications. Client machine includes a plurality of input devices which in a present embodiment includes a keyboard 200 and a microphone 204. Other input devices, such as a touch screen, and camera lens are also contemplated. Input from keyboard 200 and microphone 204 is received at a processor 208, which in turn communicates with a non-volatile storage unit 212 (e.g. read only memory ("ROM"), Erasable Electronic Programmable Read Only Memory ("EEPROM"), Flash Memory) and a volatile storage unit 216 (e.g. random access memory ("RAM")).

[0055] Programming instructions that implement the functional teachings of client machine 54 as described herein are typically maintained, persistently, in non-volatile storage unit 212 and used by processor 208 which makes appropriate utilization of volatile storage 216 during the execution of such programming instructions. Of particular note is that non-volatile storage unit 212 persistently maintains a native menu application 82 and a web-browser application 86, each of which can be executed on processor 208 making use of volatile storage 216 as appropriate. Various other applications (not shown) are maintained in non-volatile storage unit 212 according to the desired configuration and functioning of client machine 54, one specific non-limiting example of which is a contact manager application 90 which stores a list of contacts, addresses and phone numbers of interest to user U and allows user U to view, update, delete those contacts, as well as providing user U an option to initiate telecommunications (e.g. telephone, email, instant message, short message service) directly from that contacts application.

[0056] Native menu application 82 is configured to provide menu choices to user U according to the particular application (or other context) that is being accessed. By way of example, while user U is activating contact manager application 90, user U can activate menu application 82 to access a plurality of menu choices available that are respective to contact manager application 90. This example is shown in greater detail in Figure 3. In Figure 3, a non-limiting exemplary portion of keyboard 200 is shown, which comprises a menu key 232, a pointing device in the form of a trackball 236, and a select key 240. Figure 3 also shows a non-limiting example of how contact manager application 90 can be rendered on display 224

when being accessed. In Figure 3, contact manager application 90 is shown displaying two contacts and a telephone number for each, namely, Bill Smith at 555-555-1212 and Sally Struthers at 555-555-1313. Note that Sally Struthers is highlighted using a colour scheme that is inverse to the colour scheme used to display “Bill Smith” and the words “Contact Manager Application”, indicating that Sally Struthers is currently being selected. User U can operate trackball 236 to scroll between Bill Smith and Sally Struthers causing one or the other to be highlighted.

[0057] While accessing contact manager application 90 as shown in Figure 3, user U can also depress menu key 232 which will invoke menu application 82, an exemplary result of which is shown in Figure 4. Menu application 82 provides a contextual menu M-90 comprised of a plurality of menu choices that are reflective of the context in which menu key 232 was selected. (Contextual menu M-90 in Figure 4 is respective to contact manager application 90 and hence the suffix “-90” in M-90. Generically, however, contextual menus will be referred to herein as contextual menus M.) In the example in Figure 4, contextual menu M-90 provides the choices of: “Help” to obtain context sensitive help about what options are available to user U within the contact manager application 90; “View” to allow user U to see more contact information (e.g. address, additional phone numbers, photographs) of the highlighted contact; “Edit” to allow user U to edit the same information that can be viewed using “View”; “Delete” to allow user U to delete the particular contact from non-volatile storage memory; “Call” to allow user U to invoke a telephony application to initiate a telephone call to the highlighted contact; “Email” to allow user U to invoke an email application to compose an email to the highlighted contact; “Close” to allow user U to close contact manager application 90 altogether and return to an application selection screen (not shown). While, for example “Call Sally Struthers” is highlighted, user U can depress the select key 240 in order to cause client machine 54 to invoke a telephony application (not shown) and dial the telephone number for Sally Struthers. User U can also depress menu key 232 while menu application 82 is open to cause menu application 82 to close and return control to the contact manager application 90 in accordance with the discussion relative to Figure 3.

[0058] Note that the options in contextual menu M-90 are stored within non-volatile storage 212 as being specifically associated with contact application 90. Menu application 82 is therefore configured to generate a plurality of different contextual menus M that are reflective of the particular context in which the menu application 82 is invoked. For example, in an email application where an email is being composed, invoking menu application 82 would generate a

contextual menu M that included the options of sending the email, cancelling the email, adding addresses to the email, adding attachments, and the like. The contents for such a contextual menu M would also be maintained in non-volatile storage 212. Other examples of contextual menus M will now occur to those of skill in the art. Menu application 82 and contextual menus M will be discussed in greater detail below.

[0059] Returning now to Figure 1, web-server 58 and schema server 62 (which can, if desired, be implemented on a single server) can be based on any well-known server environment including a module that houses one or more central processing units, volatile memory (e.g. random access memory), persistent memory (e.g. hard disk devices) and network interfaces to allow servers 58 and 62 to communicate over network 66. For example, server 58 or server 62 or both can be a Sun Fire V480 running a UNIX operating system, from Sun Microsystems, Inc. of Palo Alto Calif., and having four central processing units each operating at about nine-hundred megahertz and having about sixteen gigabytes of random access memory. However, it is to be emphasized that this particular server is merely exemplary, and a vast array of other types of computing environments for servers 58 and 62 are contemplated.

[0060] It should now be understood that the nature of network 66 and the links 70, 74 and 78 associated therewith is not particularly limited and are, in general, based on any combination of architectures that will support interactions between client machine 54 and servers 58 and 62. In a present embodiment network 66 itself includes the Internet as well as appropriate gateways and backhauls to links 70, 74 and 78. Accordingly, the links 70, 74 and 78 between network 66 and the interconnected components are complementary to functional requirements of those components.

[0061] More specifically, system 50 includes link 70 between client machine 54 and network 66, link 70 being based in a present embodiment on core mobile network infrastructure (e.g. Global System for Mobile communications ("GSM"); Code Division Multiple Access ("CDMA"), Enhanced Data rates for GSM Evolution ("EDGE"), Evolution Data-Optimized ("EV-DO"), High Speed Downlink Packet Access ("HSPDA").) or on wireless local area network ("WLAN") infrastructures such as the Institute for Electrical and Electronic Engineers ("IEEE") 802.11 Standard (and its variants) or Bluetooth or the like or hybrids thereof. Note that in an exemplary variation of system 50 it is contemplated that client machine 54 could be other types of client machines, including a full desktop computer or a "thin-client".

[0062] System 50 also includes link 74 which can be based on a T1, T3, O3 or any other

suitable wired or wireless connected between server 58 and network 66. System 50 also includes link 78 which can be based on a T1, T3, O3 or any other suitable wired or wireless connected between server 62 and network 66.

[0063] As previously stated in relation to Figures 1 and 2, client machine 54 is configured to interact with content available over network 66, including web content on web-server 58. In a present embodiment, client machine 54 effects such interaction via web-browser application 86 that is configured to execute on client machine 54. As will be explained further below, web-browser application 86 is a mini-browser in the sense that it is configured to render web-pages on the relatively small display 224 of client machine 54, and during such rendering attempt to render those pages in a format that is different from how those pages would be rendered on a traditional desktop browser, but still conveys, as much as possible, substantially the same information as if those web-pages had been rendered on a full browser such as Internet Explorer or Firefox on a traditional desktop or laptop computer. Web-server 58 is configured to host a web-site 100 that includes a plurality of web-pages.

[0064] Figures 5-8 show exemplary representations of four different pages from web-site 100, labeled 100-1, 100-2, 100-3 and 100-4 respectively. The representation in Figures 5-8 shows how web-pages 100-1, 100-2, 100-3 and 100-4 would be rendered on a traditional desk-top computer such as a Windows-based computer running the Internet Explorer or Firefox Web-browser as an HTTP web-page. In the example, web-site 100 is an e-commerce web-site belonging to a fictional computer equipment retailer named ABC ComTech Corp. Web-site 100 can be browsed to select various computer equipment items for purchase, culminating in the selection of a secure checkout screen that can be used to complete the final order for the selected computer equipment and to provide payment and shipping information therefor. Figures 5-8 shows exemplary navigation using a traditional desk-top browser through the "Home"; "Computers"; "Laptops" and "17.0 inch" menu options as found in the menu-panes indicated at 104-1, 104-2, 104-3 and 104-4 respectively on Figures 5, 6, 7, and 8. Figures 5-8 also show content panes indicated at 108-1, 108-2, 108-3 and 108-4 respectively on Figures 5, 6, 7, and 8. In the exemplary pages on Figures 5-8, it will be noted that content panes 108-1, 108-2, 108-3, 108-4 comprise promotional content that corresponds to the level of the menu in its respective menu-panes 104-1, 104-2, 104-3, 104-4. More particularly, in Figure 5, which corresponds to the "Home" menu-pane 104-1, there are promotional items (e.g. A camera, a computer, a personal navigation device, and a television) presented in content pane 108-1 that reflect more than one of the options in the "Home" menu-pane 104-1. In Figure 6, which

corresponds to the “Computers” menu-pane 104-2, there are promotional items (e.g. a laptop computer and a desktop computer) presented in content pane 108-2 that reflect more than one of the options in the “Computers” menu-pane 104-2. In Figure 7, which corresponds to the “Laptops” menu-pane 104-3, there are promotional items (e.g. various laptop computers) presented in content pane 108-3 that reflect more than one of the options in the “Laptops” menu-pane 104-3. In Figure 8, which corresponds to the 17.0” laptops page, menu-pane 104-4 is substantially the same as menu-page 104-3, while content pane 108-4 includes a list of 17” laptops that are available for purchase. (Note that the fact that menu-pane 104-4 and menu-pane 104-3 are the same is purely exemplary and that in general menu-panes can contain any desired menu selections.) If desired, a specific laptop listed on content pane 108-4 can be selected for further information and/or selected for purchase.

[0065] Those skilled in the art will now recognize that menu-panes 104-1, 104-2, 104-3 and 104-4 represent at least one set of hyper-text markup language (“HTML”) programming instructions possibly incorporating scripting language such as Java-script. Likewise those skilled in the art will now recognize that content-panes 108-1, 108-2, 108-3 and 108-4 represent at least one other set of hyper-text markup language (“HTML”) programming instructions possibly incorporating scripting language such as Java-script. The programming instructions for menu-panes 104-1, 104-2, 104-3 and 104-4 are discrete from the programming instructions for content-panes 108-1, 108-2, 108-3 and 108-4. It will also now be apparent that, web-server 58 is configured to provide each web-page 100-1, 100-2, 100-3 and 100-4 in its entirety in response to a request from a web-browser, so that it is not generally possible to view, for example web-page 100-4 directly from web-page 100-1 or web-page 100-2.

[0066] Referring again to Figures 1 and 2, in a present embodiment, web-browser application 86 is also configured to interact with schema server 62 in order to obtain a schema 102. In general, a schema such as schema 102 comprises a file corresponding to content on web-site 100. Such a schema file can be generated in any desired format, such as eXtensible Markup Language (“XML”) or a text file. A schema can contain instructions to identify each page family on the website as well as instructions to extract desired objects and elements for each page family. A schema can additionally specify the relationship between the objects and attributes. In a present embodiment schema 102 includes information relative to menu-panes 104-1, 104-2, 104-3 and 104-4 that is usable to menu application 82 and web-browser application 86 in the presentation of web-site 100 on client machine 54. Table I shows an exemplary representation of a schema 102 that corresponds to web-site 100.

TABLE I
Exemplary content of schema 102 corresponding to exemplary web-site 100

<u>Root Menu Item</u>	<u>Level 1 Menu Item</u>	<u>Level 2 Menu Item</u>	<u>Level 3 Menu Item</u>	<u>Web-page Link within web-site 100 containing content 108 to be shown corresponding to Menu</u>
Home		N/A	N/A	Address 0 (Corresponds to Web-page 100-1)
	Computers	N/A	N/A	Address 1 (Corresponds to Web-page 100-2)
	Computers	Laptops	N/A	Address 2 (Corresponds to Web-page 100-3)
	Computers	Laptops	13.3" and smaller	Address 3 (Corresponding web-page not shown)
	Computers	Laptops	14.1"	Address 4 (Corresponding web-page not shown)
	Computers	Laptops	15.4"	Address 5 (Corresponding web-page not shown)
	Computers	Laptops	Refurbished laptops	Address 6 (Corresponding web-page not shown)
	Computers	Laptops	Tablet and speciality	Address 7 (Corresponding web-page not shown)
	Computers	Laptops	17.0"	Address 8 (Corresponds to Web-page 100-4)
	Computers	Desktop Computers	N/A	Address 9 (Corresponding web-page not shown)
	Computers	Desktop Computers	(Model Line 1)	Address 10 (Corresponding web-page not shown)
	Computers	Desktop Computers	(Model Line 2)	Address 11 (Corresponding web-page not shown)

	Computers	Desktop Computers	(Model Line 3)	Address 12 (Corresponding web-page not shown)
	Computers	Monitors	N/A	Address 13 (Corresponding web-page not shown)
	Computers	Monitors	(Model Line 1)	Address 14 (Corresponding web-page not shown)
	Computers	Monitors	(Model Line 2)	Address 15 (Corresponding web-page not shown)
	Computers	Monitors	(Model Line 3)	Address 16 (Corresponding web-page not shown)
	Computers	Computer Packages	N/A	Address 17 (Corresponding web-page not shown)
	Computers	Computer Packages	(Model Line 1)	Address 18 (Corresponding web-page not shown)
	Computers	Computer Packages	(Model Line 2)	Address 19 (Corresponding web-page not shown)
	Computers	Computer Packages	(Model Line 3)	Address 20 (Corresponding web-page not shown)
	Computers	Apple Computers	N/A	Address 21 (Corresponding web-page not shown)
	Computers	Apple Computers	(Model Line 1)	Address 22 (Corresponding web-page not shown)
	Computers	Apple Computers	(Model Line 2)	Address 23 (Corresponding web-page not shown)
	Computers	Printers and Fax Machines	N/A	Address 24 (Corresponding web-page not shown)
	Computers	Printers and Fax Machines	(Model Line 1)	Address 25 (Corresponding web-page not shown)
	Computers	Printers and	(Model Line 2)	Address 26 (Corresponding web-

		Fax Machines		page not shown)
	Computers	Scanners	N/A	Address 27 (Corresponding web-page not shown)
	Computers	Scanners	(Model Line 1)	Address 28 (Corresponding web-page not shown)
	Computers	Scanners	(Model Line 2)	Address 29 (Corresponding web-page not shown)
	Computer Add-ons	N/A	N/A	Address [n] (Corresponding web-page not shown)
		[Sub-levels for Computer Add-ons Per above structure]		
	Software	N/A	N/A	Address [n1] (Corresponding web-page not shown)
		[Sub-levels for Software Per above structure]		
	Photo	N/A	N/A	Address [n2] (Corresponding web-page not shown)
		[Sub-levels for Photo Add-ons Per above structure]		
	Photo-Finishing	N/A	N/A	Address [n3] (Corresponding web-page not shown)
		[Sub-levels for Photo-Finishing Add-ons Per above structure]		
	TV & Video	N/A	N/A	Address [n4] (Corresponding web-page not shown)
		[Sub-levels for TV& Video Add-ons Per above structure]		
	Audio	N/A	N/A	Address [n5] (Corresponding web-page not shown)
		[Sub-levels for Computer Add-ons Per above structure]		

[0067] Explaining Table I in greater detail, the first four columns of Table I (“Root Menu Item”; “Level 1 Menu Item”; “Level 2 Menu Item”; “Level 3 Menu Item”) correspond to the menu structure found in menu-panes 104-1, 104-2, 104-3, 104-4. The last column of Table I (“Web-page Link within web-site 100”) corresponds to the specific address associated with a

particular web-page within website 100, including web-pages 100-1, 100-2, 100-3, 100-4 and other web-pages that are not actually shown in the Figures and points to the respective content (including 108-1, 108-2, 108-3, 108-4 and other content not actually shown in the Figures) that is associated with the menu-panes reflected in the associated first four columns. Thus the first four columns can be used by native menu application 82 to create a plurality of contextual menus M that have substantially the same content as menu-panes 104-1, 104-2, 104-3 and 104-4. Likewise, the last column of Table I can be used to extract web-content corresponding to the web-site address indicated in the relevant entry of that last column, as found within web-site 100, including web-content 108-1, 108-2, 108-3, 108-4 and other web-content from other web-pages in web-site 100 that are not actually shown in the Figures. Web-browser application 86 and native menu application 82 are therefore configured to co-operate using schema 102 in order to present web-content within the web-browser application 86, while using native menu application 82 to permit user U to navigate through web-site 100.

[0068] Referring now to Figure 9, a method for content navigation is represented in the form of a flow-chart as indicated generally at 900. Method 900 can be performed using system 50, though it is to be understood that method 900 can be performed on variations of system 50, and likewise it is to be understood that method 900 can be varied to accommodate variations on system 50.

[0069] At block 910 a schema is requested. Block 910 is performed by web-browser application 86 (or a separate plug-in or other application configured to execute in conjunction with web-browser, such as a transcoding engine, not shown) which establishes a connection with schema server 62 in order to retrieve schema 102. At block 915 the schema is validated and returned. The validation of block 915 (which, it will be appreciated, like certain other aspects of method 900, will be understood to be optional) can be effected by server 62 which can perform a validation operation to confirm that schema 102 matches web-site 100 and is otherwise up-to-date. If validation is not achieved then an exception (e.g. an error) can be generated. Assuming validation is achieved, then schema 102 is returned to web-browser application 86 where it is loaded into web-browser application 86. Blocks 910 through 915 are represented in Figure 10, as a connection between web-browser application 86 of client machine 54 and schema 102 of server 62 is indicated at reference 316 such that schema 102 is now loaded onto client machine 54 and available to web-browser application 86.

[0070] Also note that the means by which web-browser application 86 requests schema 102 is

not particularly limited. In one particular embodiment, however, it is contemplated that web-browser application 86 will be configured to automatically make network requests over network 66 to request a schema that corresponds to website 100. For example, schema server 62 can have a predefined network address on network 66 that is preprogrammed into client machine 54. The type of network address is not particularly limited, and can be, for example, any type of network identifier such as an Internet Protocol ("IP") address or a Uniform Resource Locator ("URL"). Any other suitable type of network address is contemplated. Client machine 54 can therefore be programmed to send a request to the address for schema server 62 and request that schema server 62 provide, if available, a schema (e.g. schema 102) that corresponds to web-site 100. (Note of course that in other embodiments, a separate schema can be provided for each web-page within web-site 100). The request at block 910 provided by client machine 54 can be formed with any unique identifier for each web-page, but in the context of the Internet the request would most typically be, or derived from, the URL associated with each web-page. In turn, that unique identifier can be used to index schema 102 on schema server 62.

[0071] As well, authentication can be made through connection 316 to validate the origin of schema 102. For an example, private and public key based authentication can verify that schema 102 is originated from a trusted source.

[0072] Those skilled in the art will now recognize that system 50 can be implemented so that a plurality of web-sites (like web-site 100) are hosted over network 66 (either alone by server 58 or by a plurality of web-servers like web-server 58), and that a corresponding plurality of schemas for each of those web-sites (or each of the web-pages therein, or both) can be maintained on schema server 62. Those skilled in the art will now recognize that there can in fact be a plurality of schema servers (like schema server 62) and that client machine 54 can be configured to search for corresponding schema files on one or more of those schema servers. Those skilled in the art will now further recognize that schema servers can be hosted by a variety of different parties, including, for example: a) a manufacturer client machine 54, b) a service provider that provides access to network 66 via link 70 on behalf of user U of client machine 54; or c) the entity that hosts web-site 100. In the latter example it can even be desired to simply host schema 102 directly on web-server 58 and thereby obviate the need for schema server 62.

[0073] Referring again to Figure 9, at block 920 a web-page is selected. In this case home

web-page 100-1 for web-site 100 is selected. Such a selection will typically have been made as part of web-browsing performed by user U, and indeed will have been done prior to invocation of method 900. In this embodiment, web-browser application 86 makes a request for home web-page 100-1. Such a request can be made directly bypassing server 62 altogether. (In other embodiments, discussed below in relation to Figure 17, such a request can be made via server 62 or another server, with intermediate transcoding (e.g. transcoding of web-content 108-1, 108-2, 108-3 and 108-4) from the format of that content on web-server 58 into another format that is optimized for generation on display 224). At block 925, the selected web-page is requested, and at block 930 the selected web-page is returned. More particularly, web-server 58 returns web-page 100-1 to web-browser application 86. Blocks 920 through 930 are represented in Figure 11 as a connection between web-browser application 86 and web-server 58 is indicated at 220 such that web-page 100-1 is now loaded onto client machine 54 and available to web-browser application 86.

[0074] Referring again to Figure 9, at block 935 the web-page is generated using the schema within the web-browser. Block 935 is represented in Figure 12, as In this example, web-page 100-1 is generated on display 224 using the last column of Table I representing the aspect of schema 102 that corresponds with the home web-page 100-1 of web-site 100. As a result of performance of block 935, only content 108-1 is actually shown on display 224 while menu-pane 104-1 is removed from display within web-browser application 86.

[0075] At block 940, a determination is made as to whether native menu application 82 has been selected for activation. In a present embodiment, and referring again to Figure 12, such a determination would be made by determining whether menu key 232 had been depressed. A yes determination would be made at block 940 if key 232 was depressed, whereupon method 900 would advance to block 945. If key 232 is not depressed, then a no determination would be made and method 900 would cycle back to block 935.

[0076] Within block 935, user U can perform the usual functions of web browsing, including scrolling through the page, and selecting any individual links which may be active on within content 108-1. Thus, user U could browse and otherwise interact with content 108-1 as if user U was operating a traditional desktop browser. It will now be understood that such interaction could lead to a selection of a different web-page which would otherwise interrupt performance of method 900. Such interaction is not contemplated by method 900 expressly for convenience and simplicity, but that is not to say that such interaction is excluded.

[0077] Assuming, however, that a “yes” determination is made at block 940 and method 900 advances to block 945, then at block 945 a contextual menu would be generated. Figure 13 represents performance of block 945, as invocation of menu application 82 has caused contextual menu M-104-1 to be rendered in conjunction with content 108-1 on display 212. As described above, contextual menu M-104-1 is generated using native menu application 82. Native menu application 82 interacts with web-browser application 86 in order to obtain the relevant contents of menu-pane 104-1 in order to ultimately generate contextual menu M-104-1.

[0078] At block 950, a determination is made as to whether a web-page has been selected. User U can thus scroll through the various options presented on contextual menu M-104-1 in much the same manner that user U could scroll through the options presented on contextual M-90 as discussed above. Thus, at block 950, a determination would be made as to whether user U interacting with contextual menu M-104-1 using menu application 82 made a selection corresponding to one of “Computers”; “Computer Add-ons”; “Software”; “Photo-finishing”; “TV & Video”; or “Audio”.

[0079] If the determination at block 950 is “no” then at block 955 a determination is made as to whether a selection was made to close the menu application. Continuing with the present example, it would be determined whether user U interacting with contextual menu M-104-1 using menu application 82 made a selection corresponding to “Close Menu”. If the determination at block 955 is “yes”, then method 900 returns to block 935 and contextual menu M-104-1 would close and display 224 would return the appearance as shown in Figure 12.

[0080] If the determination at block 955 is “no” then method 900 advances to block 960 where a determination is made as to whether a control item was selected. Continuing with the present example, in Figure 13 an exemplary control option entitled “close browser” is provided. (It should be noted that other control options can be provided, such as “switch application”. Other control options will now occur to those skilled in the art.) A “yes” determination would therefore be made at block 960 if user U interacting with contextual menu M-104-1 using menu application 82 made the selection corresponding to “Close Browser”. Such a “yes” determination could lead to the termination of method 900, at block 965, as web-browser application 86 is closed altogether and operation of client machine 54 directed to execution of another application, such as a main menu application (not shown).

[0081] If the determination at block 960 is “no”, (i.e., user U interacting with contextual menu M-104-1 using menu application 82 made a selection corresponding to “Home”), then method 900 cycles back to block 950.

[0082] Referring again to block 950 of Figure 9, if the determination at block 950 was “yes”, (i.e. that user U interacting with contextual menu M-104-1 using menu application 82 made a selection corresponding to one of “Computers”; “Computer Add-ons”; “Software”; “Photo-finishing”; “TV & Video”; or “Audio”), then method 900 cycles back to block 925 where another web-page corresponding to the selection is made. Method 900 continues to perform thereafter in substantially the same manner as previously described, except that the newly selected web-page is now generated and the corresponding contextual menu for each of those pages loaded accordingly.

[0083] Figures 14, 15 and 16 show further exemplary screen shots of how navigation would be effected using method 900 proceeding from the screen shot in Figure 13. It will now be appreciated that Figure 13 corresponds to Figure 5; Figure 14 corresponds to Figure 6; Figure 15 corresponds to Figure 7; and that Figure 16 corresponds to Figure 8, except that Figures 5-8 show how various pages of web-site 100 would be rendered using a traditional desktop browser, whereas Figures 13-16 shows how those same pages would be rendered using method 900. Note that on Figure 16, corresponding to Figure 8, user U can, if desired, select a specific laptop listed on content pane 108-4 using browser application 86 in order to obtain further information and/or selected for purchase.

[0084] Referring now to Figure 17, a system for content navigation in accordance with another embodiment is indicated generally at 50a. System 50a is a variation of system 50 and therefore like elements in system 50a bear like references to counterpart references in system 50, except followed by the suffix “a”. Of note, however, is that in system 50a, server 62a includes a transcoding engine 103a that is configured to, on behalf of device 54a, transcode web-content 108a-1, 108a-2, 108a-3 and 108a-4 from the format of that content as maintained by web-server 58a into another format that is optimized for generation on the display of device 54a. Thus, in system 50a, web-content 108a-1, 108a-2, 108a-3 or 108a-4 destined for device 54a is retrieved from server 58a via server 62a, whereby server 62a transcodes that content prior to sending that content to device 54a. As a non-limiting example, the transcoded version of web-content 108a-1 is identified as web-content 108a'-1 in an oval associated with web-browser 86a. In this way, a transcoding engine (or other transcoding functions) need not be

placed on (or at least implemented by) device 54a and thereby freeing up resources on device 54a. Those skilled in the art will now recognize that block 935 of method 900 would be modified for system 50a, whereby the webpage would be generated using transcoding engine 103a instead of doing any transcoding using the schema within browser 86.

[0085] Referring now to Figure 18, a method for content navigation in accordance with another embodiment is represented in the form of a flow-chart as indicated generally at 900b. Method 900b can be performed using system 50, though it is to be understood that method 900b can be performed on variations of system 50, and likewise it is to be understood that method 900b can be varied to accommodate variations on system 50. Method 900b is a variation on method 900 and therefore like blocks in method 900b bear like references to counterpart blocks in method 900, except followed by the suffix "b". While methods 900 and 900b are substantially the same, of note is that in method 900b, block 945 is omitted and substituted with block 946b. Block 946b itself is a variation on block 945, except that the menu pane M-104 that is generated includes additional depth beyond the depth provided in the original menu pane 104. Additional depth is meant to indicate, for example, that when generating menu pane 104-1 on device 54, menu pane M-104 may be modified to include at least a portion of the menu selections found one or more of menu panes 104-1, 104-2, 104-3, 104-4. (An exemplary modified version of menu pane M-104b-1 generated by block 946b is shown in Figure 19 as menu pane M-104b-1, which will be discussed in further detail below.) In one exemplary extreme, not shown in the Figures, menu pane M-104 may be modified to include ALL of the selections in 104-1, 104-2, 104-3, 104-4.

[0086] Method 900b addresses one problem of browsing between web-pages on mobile electronic devices, whereby browsing through multiple pages can be time consuming, resource (e.g. bandwidth, processor, memory) intensive and not to mention financially expensive for user U depending on the rate plan available to user U. Method 900b can allow users to navigate through multiple levels of web page menus. Turning now to Figure 19, in menu pane M-104b-1, the selections of menu pane M-104-3 are combined into the selections of menu pane M-104-1, thereby allowing user U to navigate directly the contents of menu pane M-104-3 and bypassing the contents of menu pane M-104-2. As a practical example, if user U wishes to view 17.0" laptops (i.e. the content on web-page 108-4 in Figure 16), then rather than having to navigate through web-pages 108-1, 108-2, and 108-3, one page at a time, user U can be permitted to reach web-page 108-4 in Figure 16 directly from web-page 108-1 in Figure 19, directly selecting the ultimate target without having to go through each intermediary

web page 108-1, 108-2, and 108-3. Implementing method 900b can be effected by examining the full contents of Table I and generating a modified menu pane M-104 that reflects the desired combinations of one or more of menu panes M-104-1, M-104-2 and M-104-3.

[0087] The determination of which portions of menu panes M104-1, M-104-2 or M-104-3 are to be combined are not particularly limited. For example, a record can be kept of the most popular selections by all users of web site 100 and to include direct links to those selections. Alternatively, specific promotions can be chosen to be combined into the modified menu pane M-104 (e.g. where the operator of server 58 wishes to promote the sale of 17"0 laptops in Figure 16). Alternatively, a browsing history by user U of device 54 can be maintained, so that the first time user U browses web-site 100, method 900 is invoked so that user U is presented with the screens shown in Figures 13, 14, 15 and 16, but upon returning to web-site 100, method 900b is invoked and user U will be initially presented with the screen shown in Figure 19 in anticipation of user U's desire to browse directly to the screen shown in Figure 16.

[0088] The foregoing presents certain exemplary embodiments, but variations or combinations or subsets thereof are contemplated. For example, other functions can be added to each contextual menu M as those menus are presented within browser application 86, such as the common "back" or "forward" commands as found in traditional desk top browsers. Also, the types of web-sites 100 are not intended to be limited to e-commerce web-sites.

[0089] Another embodiment provides a communications environment 10D. Referring to Figure 20, a communications environment 10D has one or more Web sites 20D that have a collection of Web files 52D on a particular subject that includes a beginning file called a home page, which is reached by a computing device 101D over a communications network 11D via a network address (e.g. URL). From the home page, or through direct access to any page without going through the respective home page, a user, using a web site browser (via the device 101D), can access both content 50D and related navigation 54D of all the other pages on the Web site. It is recognized that the Web site is typically hosted on one or more Web servers. A server in this context is a computer device 101D that holds the files for one or more Web sites. For example, a large Web site may be hosted on a number of servers located in many different geographic places.

[0090] Access to the Web sites over the network 11D can be done directly, in terms of desktop devices 26D, and through a proxy gateway 22DD, further described below. Accordingly, one or more mobile devices 24D (e.g. PDAs, mobile phones, etc.) and one or more desktops 26D can

use the gateway to access the pages (both content 50D and navigational 54D aspects). The gateway can be used to format or otherwise monitor the interaction of the user of the devices 24D, 26D with the content 50D and navigational 54D aspects of the Web pages.

[0091] Overview of the Environment 10D

[0092] Specifically, the environment 10D can take unstructured webpage (e.g. HTML) and convert it into a structured database, for example. It is not about simplifying HTML for any page, it is about understanding the data in a page and the relationships (between data content and between data content and navigational items tied to that page content) that govern the data in the page. Accordingly, knowledge of the data contained in the page content (e.g. data type –navigation verses published content – as well as which of the published content is related to each other and which of the navigation data is related to each other and to which published content on the page) can be used (for example via a signature file) to extract data from the web site (for example on a page by page or other defined collection of information such as for file by file) for consumption by the mobile/desktop device 101D. Therefore, it is the gateway that acts as the proxy between the desktop/mobile for accommodating requests for web site data from the mobile/desktop and corresponding web site data sent from the web site in response to the request. It is recognized that the data (e.g. web page 60D) obtained by the gateway, from the web site, could be any structured file (e.g. an HTML, XML, etc.) document (optionally in the form of a web page), or which the signature file has predefined knowledge about the contents of the document (e.g. meaning of data contained within tags/delimiters as well as the interrelationships between the data in the document). One example of this is a web page described in HTML, which can be referred to as unstructured content.

[0093] It is recognized that the extraction process of the gateway for extracting data from the web page of the web site can be used to obtain only that data (e.g. published content and/or navigational data) that is pertinent to a simplified display on the screen of the user device 101D. The reason for generation of the simplified display of the data obtained from the original web site content (e.g. a web page) can be such as but not limited to: limited display space for the generated simplified data display on the user device 101D (e.g. physical space restrictions such as for a mobile screen or for user/system defined space restrictions such as for only a portion of the theoretically available desktop screen space; and for user preference pertaining to continuity of browsing/transactional/session experience. An example of user preference is where the user starts the interaction with the web site and resultant displayed data (published

content and navigational data) on the mobile (i.e. mobile formatted data display) and then wishes to retain the formatting of the mobile when continuing to view on the desktop screen. For example, the user on the desktop can continue to browse the published content and navigational data of the web site as previously experienced on the mobile, using only a portion of the desktop screen (for example) for data display.

[0094] The remaining description will refer to the document obtained from the web site as a web page, for exemplary purposes only. Large data-driven sites don't maintain thousands of pages. They have a few page templates and populate them from a database of information, news, shopping etc. Each template represents a family of pages. And a family of pages has objects and attributes.

[0095] Example 1: News site

[0096] Family: List Page

[0097] Objects: lists a selection of news stories

[0098] Attributes: Title, abstract and date

[0099] Family: Detail page

[00100] Objects: lists a single news story (and maybe other related stories)

[00101] Attributes: Journalist, City, Date, Title, Full Story, Image

[00102] Example 2: E-commerce site

[00103] Family: List Page

[00104] Objects: lists a selection of products

[00105] Attributes: Image, Item Name, Price, Sale Price

[00106] Family: Search Page (a specific kind of list page)

[00107] Objects: same as list page +- a few

[00108] Attributes: same as list page +- a few

[00109] There are a few families of pages that can be managed to get an entire website accessible via a signature file, further described below:

[00110] List Pages – browse by category, by search, featured products

[00111] Detail pages: A specific object details with other information on a page

[00112] Search: to enter search information

[00113] Input: To do things like enter billing information (these are typically individual pages)

[00114] Signature files

[00115] We identify the signature for each family of pages (the family template) that 1) automatically can identify a given page on a website as part of the family and 2) differentiates that family from another family of pages. Similarly each object and attribute field can have a unique signature within a family of pages that we need to identify once for the family.

[00116] A Signature file can contain numerous pieces of information, for example namely:

[00117] 1) identifying the page family

[00118] 2) identifying the objects and attributes in the page

[00119] 3) Specifying the relationship between the objects and attributes.

[00120] In the case of a document received as a file, the signature file can contain knowledge about the type of file, the objects/attributes of the file, and the relationships between the objects and attributes in the file. A further example of the web site data can be such as but not limited to news articles and RSS feeds or other information feeds (stock tickers, etc.).

[00121] Schema Engine

[00122] This component uses the signature file for a website to create content data in response to the web page request, from the mobile/desktop, efficiently on the fly and send the data to the client. The data can include web page content data and navigational data obtained from the web page as requested. Alternatively the information can be stored to start building a database of the site, optionally. The construction of this database can be saved locally to the

gateway, otherwise cached to the local storage of the user device, and/or cached /stored at the web site or third party (e.g. a search engine service used for comparison of data from different web sites).

[00123] Separation of Navigation & Content

[00124] Navigation items are on the same page as content, but it may not make sense (in situations with limited screen real estate available) to display the page in the original web page format as obtained from the website by the gateway. Schema extracts the navigational items separately to create a navigational portion of the web page. The environment can do interesting things with the separated navigational items, such as feed it to an application in the background to help improve the browsing experience or to otherwise reformat the presentation of the navigational items on the display of the mobile/desktop, in order to help with navigation and maintaining navigation context in situations with limited display space available for presentation of the web page.

[00125] Continuance of Sessions

[00126] In the environment 10D, the user can start browsing from the PC or mobile device 101D and complete a purchase on either (or otherwise continue the sessions). We can continue the session to realize benefits, such as revenue share, that could be lost if continuance of sessions was not enabled. Continuance of sessions can also give users seamless flexibility to use their PC and mobile to buy/browse things from websites and to replicate the buy/browse information.

[00127] The continuance of sessions can be facilitated by the use of rich bookmarking that is generated from the desktop tagging tool discussed below, such that the rich bookmark is created that has bookmark (e.g. a displayable link) components such as but not limited to: a URL (e.g. network address of the web site data; and identified portions of the web site data located with respect to that URL (e.g. item image, item title, description of item, text body related to item – such as an article, etc.). The portions of web site data associated with the URL (e.g. page/file name) can be considered key or otherwise memorable data preferred by the user with respect to item(s) on the URL (for example product name/price/image).

[00128] Desktop tagging tool and automatically creating signature files

[00129] This uses artificial intelligence to analyze any page in one or more ways, such as

but not limited to:

[00130] 1) delimiter (e.g. HTML tag) structure and properties; and

[00131] 2) Spatial analysis of objects located on a rendered page.

[00132] Generally main content is closer to the centre of the page, is bigger and is meant to stand out more to the user. Properties in the HTML mark up can be used to accomplish this and we have AI that can identify these properties. One embodiment is where we use the rendered page, in combination with tag analysis. One benefit is that this feature could be used to generate the signature files automatically by guessing and at least significantly speeding up creating of signature files, if not completely automated. Another use of the desktop tagging tool is to create a list of rich bookmarks for later use by the user and/or for publishing or otherwise sharing with other users. One example of this would be a list of rich bookmarks provided by one user to another user, such that the list of rich bookmarks contains URLs and associated data from one or more web sites.

[00133] Conducting a Transaction

[00134] With regards to billing and completing a transaction. A user goes through a number of pages to navigate to an item they want to buy and then must continue browsing through a number more pages to complete a checkout or transaction. The provided description of the environment 10D includes detailed explanations of the analysis and output of a requested web page. The same process can be extended to all web pages browsed from start to end to complete a transaction. It is recognized that the transaction can be such as but not limited to: browsing for and subsequent purchase of item(s); and/or browsing and subsequent saving of published content (e.g. news article), as desired.

[00135] For example, actions one through ten in Figure 44 represent a user browsing through ABC ComTech Corp. to purchase an item. The web pages representing actions one through ten are shown in Figures 35-39 respectively.

[00136] Web Sites 20D

[00137] Referring again to Figure 20, Web sites 20D have the plurality of pages 52D (e.g. defined using HTML, XML, XHTML, JavaScript and other structured definition web programming languages— e.g. based on W3C standards – e.g. WSDL). A Web site can be

provided as a Web service, which can be a software system, designed to support interoperable device 101D to device 101D interactions over the network 11D. Web services can be Web APIs that can be accessed over the network 11D, such as the Internet, and executed on the remote device 101D hosting the requested services.

[00138] For example, a Web service definition encompasses many different systems, but can refer to clients and servers that communicate XML messages that follow the SOAP-standard, via a description of the operations supported by the server e.g. in WSDL.

[00139] The composition of the Web pages can include displayed content and navigation features.

[00140] Web pages typically have both of these features on each page and will display content in the main content areas and have navigation options through menus, as shown by example in Figure 26. This web page layout is structured for access by desktop 26D browsers, where the screens are large enough to display the entire page. However most mobile 24D browsers may not have the width and height of a typical PC monitor, therefore they can be unable to display pages as they would appear on a PC browser. One approach to deal with this is to re-organize the page and wrap content around the screen. A second approach used by the WAP standard is spatially divide a page (usually vertically) into a number of pages and allow users to navigate between each page section to view a page. A third approach (through the gateway 22DD) is to functionally divide website features into separated content 50D and navigation 54D components, further described below. One example of content 50D is published content (e.g. articles, stories, news, product information, etc.), which is data that is meant to be read/listened to by the user.

[00141] Content 50D

[00142] The content can include computer files, image media, audio files, electronic documents, which are either located on/in the Web page or are otherwise accessible through navigation/requests from a particular Web page and/or Web service. For example, Web content can be referred to as textual, visual or aural content that is encountered as part of the user experience through interaction with Web sites/services. Web content may include, among other things: text; images; sounds; videos; animations; and feeds (video, audio, and/or textual). For example, the pages can present content as predominantly composed of HTML, or some variation, as well as data, applications, e-services, images (graphics), audio and video

files, personal Web pages, archived e-mail messages, and many more forms of file and data systems can belong to Web sites and Web pages.

[00143] Examples of content can be as follows:

[00144] 1) Tables for presenting information displayed in a grid, such as a calendar, or in a spreadsheet, such as financial data. Tables can be used to have greater control over page layout.

[00145] For example, a table can help that text and graphics are displayed in their correct location. A table can also encompasses an entire page, with nested tables (including content and/or navigation features) within the main table for even more layout control;

[00146] 2) video and audio files;

[00147] 3) text, e.g. articles, for most web pages, is one of the most important features. Text can be used to present ideas, instructions, and/or educational/recreational content; and

[00148] 4) Images (e.g. GIF, JPEG) can be used in web pages to support the theme of the web page and to provide a visual impression. Images can be separate image files and may not reside in the HTML document itself, but can be stored in the same location as the web page. Images can be scanned photographs or pictures, may be created in a draw program, or may be downloaded from another web site.

[00149] Navigation 54D

[00150] The mobile 24D and desktop 26D devices coordinate user events 109D of the respective users, through operation of the browsers (or other applications) 207D in interaction with the supporting navigation features of the Web pages/sites. The navigation features can include visual based controls, text controls, and/or a combination thereof. For example, there can be three basic types of navigation: Hierarchical that applies to Web sites that are information-rich and are organized as a large tree, much like a library; Global that applies to Web sites where the user can logically jump among all points (e.g. content and/or other navigation controls); and Local that applies when the user wants to access a depth of information/content within broader areas/content of the Web site.

[00151] Examples of navigation 54D mechanisms with respect to the content of a Web site can include such as but not limited to: embedded links (e.g. anyplace where one links content

within the body of the page); and navigation buttons, graphic and text-based. As well, text entry fields can be used to navigationally access content and other navigation features of Web sites.

[00152] Further examples of navigation 54D mechanisms can be such as but not limited to:

[00153] 1) Buttons can be images with text on them that provide a means to navigate from one location to another. Buttons may be created in a draw program or downloaded from other web sites.

[00154] 2) Menu Bar can be features on a web page that provide links to other pages for easy navigation between the pages or other Web sites. Menu bars may contain buttons (e.g. text/images), they may be created as a table, or they may be text-based with divider lines; and

[00155] 3) Links providing "branching capabilities" – the ability to go to another site/page. Links provide that branching option. Links are "jump starts" to other web pages/sites. A link may take the user to another page or it may take them to another site.

[00156] Devices 101D

[00157] Referring to Figure 21, a computing device 101D of the system 10D can include a network connection interface 200D, such as a network interface card or a modem, coupled via connection 218D to a device infrastructure 204D. The connection interface 200D is connectable during operation of the devices 101D to the network 11D (e.g. an Intranet and/or an extranet such as the Internet), which enables the devices 101D to communicate with each other (e.g. that of the mobile 24D, gateway 22DD, desktop 26D and Web site 20D) as appropriate. The network 11D can support the communication of the web pages as requested.

[00158] Referring again to Figure 21, the device 101D can also have a user interface 202D, coupled to the device infrastructure 204D by connection 222D, to interact with a user (e.g. mobile user, gateway administrator, website administrator, desktop user – not shown). The user interface 202D can include one or more user input devices such as but not limited to a QWERTY keyboard, a keypad, a stylus, a mouse, a microphone and the user output device such as an LCD screen display and/or a speaker. If the screen is touch sensitive, then the display can also be used as the user input device as controlled by the device infrastructure 204D.

[00159] Referring again to Figure 21, operation of the device 101D is facilitated by the device infrastructure 204D. The device infrastructure 204D includes one or more computer processors 208D and can include an associated memory 22D (e.g. a random access memory). The computer processor 208D facilitates performance of the device 101D configured for the intended task (e.g. of the respective module(s) of the host system 14D) through operation of the network interface 200D, the user interface 202D and other application programs/hardware 207D (e.g. browser or other device application on the mobile/desktop, web page – content and/or navigation – server of the gateway, and Web server of the Web site) of the device 101D by executing task related instructions. These task related instructions can be provided by an operating system, and/or software applications 207D located in the memory 22D, and/or by operability that is configured into the electronic/digital circuitry of the processor(s) 208D designed to perform the specific task(s). Further, it is recognized that the device infrastructure 204D can include a computer readable storage medium 212D coupled to the processor 208D for providing instructions to the processor 208D and/or to load/update the instructions 207D. The computer readable medium 212D can include hardware and/or software such as, by way of example only, magnetic disks, magnetic tape, optically readable medium such as CD/DVD ROMS, and memory cards. In each case, the computer readable medium 212D may take the form of a small disk, floppy diskette, cassette, hard disk drive, solid-state memory card, or RAM provided in the memory module 22D. It should be noted that the above listed example computer readable mediums 212D can be used either alone or in combination.

[00160] Further, it is recognized that the computing device 101D can include the executable applications 207D comprising code or machine readable instructions for implementing predetermined functions/operations including those of an operating system and the host system 14D modules, for example. The executable instructions 207D can be an application hosted on the user mobile/desktop for interacting with the gateway, the engine and other related components for when acting as a data proxy between the mobile/desktop and the web site, or a web service (e.g. search engine crawling tool) for use by the web site, as configured by the respective device 101D when operating within the environment 10D. The processor 208D as used herein is a configured device and/or set of machine-readable instructions for performing operations as described by example above. As used herein, the processor 208D may comprise any one or combination of, hardware, firmware, and/or software. The processor 208D acts upon information by manipulating, analyzing, modifying, converting or transmitting information for use by an executable procedure or an information device, and/or by routing the

information with respect to an output device. The processor 208D may use or comprise the capabilities of a controller or microprocessor, for example. Accordingly, any of the functionality of the executable instructions 207D (e.g. through modules associated with selected tasks) may be implemented in hardware, software or a combination of both. Accordingly, the use of a processor 208D as a device and/or as a set of machine-readable instructions is hereafter referred to generically as a processor/module for sake of simplicity.

[00161] The memory 22D is used to store data locally as well as to facilitate access to remote data stored on other devices 101D connected to the network 11D. This data can be related to data/user events of the mobile/desktop, data used by the gateway in obtaining and satisfying requests for web pages and associated content/navigation features, and/or actual Web site data, as appropriate for the use of the device 101D in the environment 10D.

[00162] The data can be stored in a table, which can be generically referred to as a physical/logical representation of a data structure for providing a specialized format for organizing and storing the data. General data structure types can include types such as but not limited to an array, a file, a record, a table, a tree, and so on. In general, any data structure is designed to organize data to suit a specific purpose so that the data can be accessed and worked with in appropriate ways. In the context of the present network environment 10D, the data structure may be selected or otherwise designed to store data for the purpose of working on the data with various algorithms executed by components of the executable instructions, depending upon the application thereof for the respective device 101D. It is recognized that the terminology of a table is interchangeable with that of a data structure with reference to the components of the network environment 10D

[00163] Example Operation Of Response 60D To Web Page Request by Gateway 22DD

[00164] Large data-driven sites 20D may not create and maintain thousands of pages. Instead, they use multiple page templates 62D and populate the templates from the database of content information. Examples would be online stores, news sites, sports information and weather. The association of the data in the database with the templates 62D is used to construct the web page(s) sent to the gateway 22DD.

[00165] See Figure 45 for an example of a request/response for a page on the web site 20D:

[00166] 1. A client makes a request to the ABC ComTech Corp.ca web server (2)

[00167] 2. The web server calls the respective page family (3) depending on the page requested

[00168] 3. The page family (3) retrieves data (all navigation and content) from the database (1) to populate data fields of the page 60D

[00169] 4. The web server (2) transmits a completed webpage 60D to the gateway 22DD at (4) It is recognized that a page family serves a specific function for use by the gateway 22DD via the signature file 64, see below. For example, ABC ComTech Corp.ca has the following families of pages:

[00170] - List Family: Displays a list of products on a page (Figure 31)

[00171] - Item Family: Displays details for a specific product (Figures 30 and 32)

[00172] - Search Family: Displays a list of product matches for a search keyword

[00173] - Other Families: Miscellaneous pages such as checkout/payment

[00174] Accordingly, the environment 10D can take advantage of the fact that each web page 50D of the website 20D can follow a recognizable pattern of content data/location and navigational items related to the content data and content location. For example, in the ABC ComTech example, 62D the text on the web page in red, located above a description of a computer product, is always the price of the computer product.

[00175] It is recognized that the web page could also be referred to as a document (e.g. file) that is analysed by the engine through use of the signature file in order to extract (or to insert in the case of passing information from the mobile/desktop back to the web site) information subset(s) of the document.

[00176] Signature File 64D

[00177] A signature file can be created once for a website and then can efficiently analyze and extract data from pages 60D from the website efficiently. One advantage is that as the signature file can be implemented as an application by the gateway 22DD, the gateway 22DD may not have to store any data from the website, and can instead fetch the data in real time

upon request as the webpage 60D matching the content/navigation request of the mobile 24D and/or desktop 26D. Another advantage of signature files is that they can be non-intrusive to the existing website infrastructure and may not require that a vendor or merchant make any changes to their website configuration/infrastructure. One preferable characteristic of the websites is the use of page families for representing the website data from their databases 22D, as further described below.

[00178] Using signature files, Mobile Applications, – rich mobile applications, can be created from large websites. Each page is “optimized” on the fly by extracting the data from the page and sending the data to a mobile device, through use of the signature file, thereby helping to significantly speed up loading time and saving bandwidth. It is recognized that the Schema Engine is also able to format the content and navigational items obtained from the web pages 60D for efficient display on the display of the mobile/desktop devices 101D. Turning unstructured page content into relational data can also be significant, and can help to enable rich features for users such as custom alerts and price comparison. A user could save an item while browsing and the Schema Engine could automatically go back every day (or other selected time period) and check to see if the item was on sale or in stock. A user could also ask to find similar items at other stores, via appropriate URL requests to the gateway, which could be a supportable feature if product information was stored, for example.

[00179] With regard to Search Engines, today, there is no way to automatically index the web and understand the specific details about unstructured information (e.g. embedded in a page format) that are resident on the web pages as embedded content and navigational items. For example a search engine index of the product page of camera A would contain all the keywords for that page and have prices - \$200, \$100, \$50, the relationship of the prices with respect to the camera is unknown to the search engine. For example, a product page can contain information about a specific product including its name, description and price but may also have other product being recommended to the user on the same page with their own prices and names. A search engine may know of all the names and prices in the page, but not know which sets of information belong to which product specifically. But the index would not know which one of those prices is the actual price of camera A. Applying signature files 64D (e.g. having knowledge of which of the content is related to other portions of the content, as well as navigational aspects of the content) to search indexes can allow search engines to unlock the value of precise content that exists in their indexes and can have the ability to significantly improve search results. It also enables new kinds of searches such as “Find the lowest price for this product” or “Show me all articles actually written by this author” which

would not show web pages that simply had the authors name in it. Accordingly, it is understood that use of web pages that are unstructured (e.g. little to no use of meta data for defining the content and navigational items resident in the document (e.g. page)), such as unstructured HTML.

[00180] With regard to Price Comparison or product recommendation sites – Price comparison sites depend on vendor submission for their offering and it can be very painful for vendors to prepare these data feeds. Crawling the sites like a search engine does not work for the reasons stated above, unless the use of a signature file is applied. Using the signature file, accurate product information could be ascertained by applying signature files to a crawled cache or index or collecting price and product information as it flows through the Schema Engine. In this case, a template of the cached/indexed information would be used to create the respective signature therefore. The information could be much richer and cover a significantly larger portion of the web more reliably and easily. A price comparison engine could automatically crawl using signature files to build a complete database of an ecommerce site, using search criteria facilitated through the signature file to implement complex searches of the web site content on a page per page basis (e.g. find all cameras with prices – done through the use of the signature file for respective web sites and then apply filters to the extracted data – e.g. identify those cameras with a price under \$200).

[00181] With respect to construction of an appropriate signature file, some terminology is explained using Figure 34 as an example:

[00182] Page family: Item Page

[00183] Object: Product (A camera)

[00184] Object Elements: Picture (1), Title (2), Price (3), Description (not shown)

[00185] A complete signature file 64D for a website 20D can contain such as but not limited to:

[00186] - Contains instructions to identify each page family on the website (list pages, search pages etc.);

[00187] - Contains instructions to extract desired objects and elements for each page family above;

[00188] - Specifies the relationship between the objects and attributes (camera has a title, picture and price);

[00189] - Captures web page navigation including navigational items and paging; and

[00190] - May enable special functionality for the website including searching, logging in a user, purchasing items etc.

[00191] The following provides an overview of constructing the various components of the signature file 64D, for use in interpreting web pages 60D obtained from the web site and for reformatting the content and navigational items of the requested web pages for use by the mobile 24D and/or desktop 26D as reformatted pages 66D. The recognition of various elements of the web pages for use in defining the signature file 64D can be obtained through manual/automated/semi-automated analysis of the web pages (content and navigation), as desired.

[00192] Identifying a page family

[00193] An identifier for a page family can meet 2 criteria:

[00194] 1) It is present in all pages belonging to the family

[00195] 2) It is NOT present in pages belonging to any other family

[00196] In one embodiment, a string identifier is used that meets the above criteria as shown in the example below.

[00197] Code snippet from webpage shown in Figure 30

...	
1	<tr>
2	<td colspan="2" class="product-details-prd-title">Acer Aspire AMD Turion 64 X2 Dual Core TL-52 1.60GHz Laptop (AS9300-5383F) - French - FS Exclusive</td>
3	<td class="product-details-r-bdr"> </td>
4	</tr>
...	

[00198] Code Snippet from webpage shown in Figure 32

...	
1	<tr>
2	<td colspan="2" class="product-details-prd-title">Sony 7.2MP Digital Camera (DSCW55B) - Black</td>

3	<td class="product-details-r-bdr"> </td>
4	</tr>
...	

[00199] The pages in Figures 30 and 32 are both from the item family. The text “product-
details-r-bdr” occurs in both pages and in fact in all pages from the item family in ABC
ComTech Corp. The text does not occur in the page shown in Figure 27 belonging to the list
family and in fact does not occur in any other family of pages other than the item page. Since
this satisfies both conditions of being a page family identifier, the highlighted text is chosen as
the identifier for the item page family.

[00200] Setting a limit

[00201] Unique reference can be defined by setting a limit on portion of a webpage.

[00202] Code snippet A from webpage shown in Figure 30.

```

1 function imageFoundHtml()
2 {
3     document.getElementById('largeImageRef').style.display="inline";
4     document.getElementById('largeImageNoRef').style.display="none";
5 }
6 //-->
7 </script>
8 <title>Future Shop: Computers, Laptops, Acer Aspire AMD Turion 64 X2 Dual Core TL-52 1.60GHz
Laptop (AS9300-5333F) - French - FS Exclusive</title>
9 </head>

```

[00203] Code snippet B from webpage shown in Figure 30

```
1 <DIV id="largeImageRef" style="display:none">
2 <a href="#"
onClick="openWindowAdv:http://www.futureshop.ca/popup/largeimagepopup.asp?logon=&langid=EN&file=/multimedia/products/large/10086374.jpg
&title=Acer+Aspire+AMD+Turion+R4+X2+Dual+Core+TL%2D52+1%2E60GHz+Laptop+%26AS9300%2D5383F%29+%2D+French+%2D+FS+Exclusive
550_524_0_0_0_0_0_0" title="Click here for larger view"></a>
3 <a href="#"
onClick="openWindowAdv:http://www.futureshop.ca/popup/largeimagepopup.asp?logon=&langid=EN&file=/multimedia/products/large/10086374.jpg
&title=Acer+Aspire+AMD+Turion+R4+X2+Dual+Core+TL%2D52+1%2E60GHz+Laptop+%26AS9300%2D5383F%29+%2D+French+%2D+FS+Exclusive
550_524_0_0_0_0_0_0" title="Click here for larger view">
4 </DIV>
```

[00204] The string “largeImageRef” is the string identifier used to identify and extract the product image for the page shown in Figure 30. Code snippets A and B above illustrate a common problem that can occur. The string identifier needed occurs previously in the document and is therefore an ambiguous identifier on the page. One solution to this problem is

constraining the scope of the Schema Engine to the appropriate part of the page in order to effectively use an identifier. This method allows the definition of seeming uniquely identifiers even if they appear elsewhere on the page.

[00205] Extracting objects and elements in a page family

[00206] The example page in Figure 34 contains a camera object with the elements Picture (1), Title (2), Price (3), Description (not shown). The signature file therefore can have instructions to identify and extract the elements above as part of the product object for all pages in the item page family.

[00207] As an example let us try to construct an instruction to identify and extract the title from any item page such as the pages shown in Figures 33 and 34. We know that the output of the instruction should be the title of the product in Figure 33: "Acer Aspire...Exclusive". Below is a code snippet around the title code of the page:

```

1      <tr>
2          <td colspan="2" class="product-details-prd-title"><span class="tx-heading3-dgrey">Acer Aspire AMD
Turion 64 X2 Dual Core TL-52 1.60GHz Laptop (AS8300-5383F) - French - FS Exclusive</span></td>
3          <td class="product-details-r-bdr">&nbsp;</td>
4      </tr>

```

[00208] The following instructions will result in the output of the title:

[00209] 1. Locate the string "product-details-prd-title"

[00210] 2. Extract the value after the string in (1) and in between the strings "<span" and "";

[00211] 3. Strip all mark up tags - "class="tx-heading3-dgrey">"

[00212] 4. The resulting string is the product's title

[00213] The code snippet for the page shown in figure 34 is below. Since both pages belong to the same family, we should be able to follow the instructions above and extract the title of this product. Notice that the instructions work and produces the string "Sony 7.2MP ...Black" which is the title of the product. A signature file specifies the instruction above as a single command with parameters.

```

1      <tr>
2          <td colspan="2" class="product-details-prd-title"><span class="h3 heading3-dgrey">Sony 7.2MP Digital
Camera (DSCW568) - Black</span></td>
3          <td class="product-details-r-prd">&nbsp;</td>
4      </tr>

```

[00214] The command representing instructions 1-4 above is shown below in a query language (e.g. the individual file entries) used in signature files developed for the purpose of data extraction, with some relevant parameters highlighted:

[00215] `<lookup type="pex" action="get_string" name="title" ref="product-details-prd-title" location="after" start="<span" end="</span>" include_sz="1" strip_tags="1" />`

[00216] The signature file and processing of signature files by the Schema Engine are discussed in more detail later.

[00217] Identifying object and element relationships

[00218] The object and element relationships can be implicitly or explicitly specified. For example in the ABC ComTech Corp. list page shown in Figure 31,, the instructions are to first identify and extract the picture, second to identify and extract the title, third the link and fourth the price. The instructions then repeat for the rest of the products on the page. The specific ordering and grouping of the instructions above implicitly define objects that consist of those elements.

[00219] Other Aspects

[00220] The example and information demonstrates how to capture data and relationships of objects and elements within a page of a web site 20D. The platform can actually capture relevant attributes of an object across pages. For example, if a user of the mobile 24D clicked through a number of pages in the following categories in ABC ComTech Corp. to get to a specific TV -SONY456: e.g. TV & Video > 19"-21" TVs > LCD TVs > SONY456.

[00221] Another aspect is the ability to capture the information across the navigation of pages about the product. In doing that, one can capture the categorization of the TV "TV & Video > 19"-21" TVs > LCD TVs >" and add that as another attribute of the object. This example shows how capturing of navigation metadata or information across pages can be a source of valuable information.

[00222] Although this example covers only displaying content, the same concepts apply for a page that requires input. The key input fields and values (e.g. the ability to enter search strings) can be identified in the same way and presented to the user of the mobile 24D and value captured and sent back to the website 20D via the gateway 22DD. The signature file 64D can be written in an xml based query language syntax (or other structured definition language and/or script language, for example) to specify the above identifiers and actions such as traversing backwards, forwards and extracting values. The language can be a SQL type query language and can be built on top of regular expressions.

[00223] Automatic Generation of Signature Files 64D

[00224] Described is a method of creating signature files that identify and extract specific contents from a webpage. It is recognized that, in view of Figure 21, a device 101D for implementing the automated method of signature file generation can host a corresponding tool with associated modules including a graphical user interface module to allow selecting contents on a page easier for a user, for example.

[00225] The contents may be navigational items, lists, specific items from a list, and other content, for example. The reason that this is useful is that signature files can be manually created, which can be time consuming, and subject to human error. Therefore by automating this process, the turn around time for interpreting a website as a database through the gateway 22DD can be substantially faster and more accurate.

[00226] The automated generation method is to break down the html document (of other format of the web pages) into a hierarchy of tags (delimiters pertaining to a schema of the definition of the pages). The resulting structure can be a tree, which defines the parent, siblings and children of each object. The process (described in the following section) can identify the key objects that contain the data required for the signature file. Once an object is identified as being a required field within the database, the object would then identify its uniqueness by examining its properties (for example class, style, id). If the object is a text node of the tree (or other hierarchical structure), the object will use the properties of its parent. If the properties of the object are not unique, then the object would expand its uniqueness to its parent, siblings and children. The process would expand in all directions uniformly (i.e. examine parent, then previous sibling, then next sibling, then first child. The properties of each of these items would also merge with the required object. This process would then be repeated on the parent, then the previous sibling, etc, until a unique identifier was found. Once

a unique identifier was found, an expression would be created for the signature. Note that at least two pages of the same family can be used to create the expression.

[00227] The user will enter the required fields to be extracted from the page. These fields can be specified by a user using a corresponding graphical user interface of the device 101D to select fields. Alternatively a tool similar to the Desktop tool (see below) could be used to automatically guess at the fields on a page. To automatically generate the signature file assumes that one knows where the key information that resides on the page (i.e. location within the document) – e.g. price, image, description, etc. For example, knowledge of where the key information (e.g. here is the image between these tags to identify the content) is located in the web page can be done using a number of methods, such as but not limited to: look at code of the page by hand to identify the tags used to indicate content type (e.g. navigation, navigation of which content, title, price, image, item description, etc.; semi-automated using a graphical tool to

highlight portions on the page and therefore visually select which content data corresponds to what meaning and other content data; and/or the use of the user assisted identification with confirmation/correction by user (further described below with the use of assisted generation that is applicable also to generation of rich bookmarks).

[00228] Example HTML document of Web page

[00229] **Item1.html**

[00230] <html>

[00231] <head></head>

[00232] <body>

[00233]

[00234] <div class="product">

[00235] <h1>Product title</h1>

[00236] <h2>Product Manufacturer</h2>

[00237]

[00238]

[00239] List Price: \$99.99

[00240]

[00241] Our Price: \$79.99

[00242]

[00243] <p>

[00244] This is a description for Product title made by Product Manufacturer

[00245] </p>

[00246] </div>

[00247] </body>

[00248] </html>

[00249] **Item2.html**

[00250] <html>

[00251] <head></head>

[00252] <body>

[00253]

[00254] <p>

[00255] disclaimer

[00256] </p>

[00257] <div class="product">

[00258] <h1>Sample title</h1>

[00259] <h2>Sample Manufacturer</h2>

[00260]

[00261]

[00262] List Price: \$109.33

[00263]

[00264] Our Price: \$99.99

[00265]

[00266] <p>

[00267] This is a description for Sample title made by Sample Manufacturer

[00268] </p>

[00269] </div>

[00270] </body>

[00271] </html>

[00272] Assumptions: The required fields are identified prior to this process either by the user or using an automated tool (such as the schema desktop tool). They can be as follows:

Item1	
Image	Product_image.gif
Title	Product Title
Price	\$79.99
List Price	\$99.00
Description	This is a description for Product title made by Product Manufacturer

Item2	
-------	--

Image	Sample_image.gif
Title	Sample Title
Price	\$99.99
List Price	\$109.33
Description	This is a description for Sample title made by Sample Manufacturer

[00273] It is recognized that different modules of the automated generation process can implement the following steps (embodied as executable instructions 207D – see Figure 21).

[00274] Step 1 – Identify the Image

[00275] From the Item1 the object `` is selected. It identifies `src` as an attribute and scans the source of item1 for `src="sample_image.gif"`. It does not find a match, so it then scans item2. If a match is found, and the matching object contained the image identified for item2, the attribute would be used to create a signature file image property. However, the item is not found in Item2, so no match has been made. Next the element looks at "`<img` " within list 1. It determines that it is the second match. When looking at Item2, the second image also provides the object that contains the image. Now that we have the matching object, we apply a similar heuristic to locate the result from within the object. If the object is a text node, the process is complete. Otherwise, the start and end of the object needs to be located. Using pattern recognition techniques, we find that the ``src=""` and that `""` ends the string. Therefore the following entry would be added to the signature file `<lookup type="pex" action="get_string" name="image" ref="`

[00276] Step 2 – Identify the Title

[00277] From the Item1 the object `<h1>Product title</h1>` is selected. It identifies that it is a text node, and uses its parent to identify uniqueness. There are no attributes for the parent `<h1>`. Next the element looks at "`<h1`" within list 1. It determines that it is the only match. When looking at Item2, there is only one match, and the matching element contains the title. Now that we have the matching object, we apply a similar heuristic to locate the result from within the object. Since the object is a text node, the process is complete. Therefore the following entry would be added to the signature file `<lookup type="pex" action="get_string" name="title" ref="<h1 start=" src=">" end="<" />`

[00278] Step 3 – Identify the Price

[00279] From the Item1 the object `<strong> $79.99 </strong>` is selected. There are no attributes to be checked for this element. Next the element looks at “<strong” within list 1. It determines that it is the second match. When looking at Item2, the second strong tag also provides the object that contains the price. Since the object is a text node, the process is complete. Therefore the following entry would be added to the signature file

```
<lookup type="pex" action="get_string" name="price" ref="<strong " repeat_ref="1" start="&gt;"
end="&lt;" />
```

[00280] Step 4 – Identify the List Price

[00281] From the Item1 the object `<strong> $99.99 </strong>` is selected. There are no attributes to be checked for this element. Next the element looks at “<strong” within list 1. It determines that it is the first match. When looking at Item2, the first strong tag also provides the object that contains the price. Since the object is a text node, the process is complete. Therefore the following entry would be added to the signature file

```
<lookup type="pex" action="get_string" name="price" ref="<strong" start="&gt;" end="&lt;"/>
```

[00282] Step 5 – Identify the Description

[00283] From the Item1 the object `<p>`, this is a description for Sample title made by Sample Manufacturer `</p>` that is selected. There are no attributes to be checked for this element. Next the element looks at “<p” within list 1. It determines that it is the first match. When looking at Item2, the first p tag does not provide the object that contains the description. The parent object `<div class="product">` is selected next. It identifies the attribute `class="product"`, and scans item1, and determines that it is the only match. The `<p` tag is processed again, limiting its search to the parent. The `<p` tag is identified as the first instance within the parent. Next the same process is performed on item2. First the attribute `class="product"` is located. The first `<p` tag that is a child of the object containing `class="product"` is found. The `<p` object also contains the description. Since the object is a text node, the process is complete. Therefore the following entry would be added to the signature file

```
<lookup type="pex" action="get_string" name="description" ref="class=&quot;product&quot;"
start="&lt;p&gt;" end="&lt;" />
```

[00284] Referring to Figure 22, accordingly, in view of the above, the automated generation methodology implemented on the comparison tool (e.g. device 101D) for the signature file 70D compares two or more delimiters (pertaining to a common schema of the definition of the pages) from each of the pages 52D in order to identify common uses of the delimiters (and their contents). Once identified as a match, the corresponding object, for example, is placed in the hierarchical structure 74D (or other ordered list, etc.).

[00285] It is recognized that the hierarchy 74D can link entities 76D either directly or indirectly, and either vertically or horizontally. The only direct links in a hierarchy, insofar as they are hierarchical, can be to the entities' immediate superior or to the entities' subordinates, although a system that is largely hierarchical can also incorporate other organizational patterns. Indirect hierarchical links can extend "vertically" upwards or downwards via multiple links in the same direction. Traveling up the hierarchy to find a common direct or indirect superior, and then down again can nevertheless "horizontally" link all parts of the hierarchy, which are not vertically linked to one another. Further, the structure 74D can also be a lists implemented using arrays or linked/indexed lists of some sort. The structure 74D can have certain properties associated with arrays and linked lists. A sequence can be another name for the structure 74D, emphasizing ordering of the entities 76D.

[00286] Further, it is recognized that the structure 74D would be represented in the signature file as the entries as noted above. It is recognized that a user of the device 101D could manually 78D amend or otherwise review the automatically generated signature file 64D, as desired.

[00287] User Assisted Generation of Signature Files 64D (Desktop tagging)

[00288] Described is a method of assisted recognition of web page contents that identifies and extracts specific contents from a web page, which could be applied in creating signature files. It is recognized that, in view of Figure 21, a device 101D for implementing the assisted method of web page content that identification and extraction can host a corresponding tool with associated modules. The recognized content could be used to provide the required fields in the signature files or could be used to create the rich bookmarks.

[00289] The web page contents may be navigational items, lists, specific items from a list, and other content, for example. The reason that this is useful is that signature files can be manually created, which can be time consuming, and subject to human error. Therefore by

helping to automate the recognition of web page contents, the turn around time for interpreting a website as through the gateway 22DD can be substantially faster and more accurate.

[00290] The following is an embodiment of the process of assisted capturing of web page contents, such as but not limited to the image, title, description, and price of a product page as shown in Figure 40. The result of the process is shown in Figure 41. The process is performed on the client side, with a call to the server (e.g. gateway 22DD and/or web site 20D). In one embodiment, the call consists of requesting a javascript. The javascript is generated dynamically on the server side. The dynamic part of the script can perform a number of functions. A first function consists of checking the users cookies for a username and password, so that the user is not prompted with a login upon saving the item. A second function uses a referrer site to load confidence intervals, that have been generated on previously saved items from the same site.

[00291] The javascript can have no other knowledge of a web-page, other than confidence intervals to determine the specific fields (image, title, description, and price) of a product, for example. The confidence intervals, further explained below, contain the location on the page (width and height) of each field, and other properties (stated below) that are used to guess a field (i.e. what is the significance/meaning of the field with respect to the content/navigation items contained on the web page. Therefore, confidence levels can be set on a per site basis, but the process used to derive the fields can be the same for every site. This can be done, because most ecommerce web sites display products in a similar fashion (e.g. the title is bold, the image is near the middle and large, the description has the most text, and is black, the price is highlighted and when rendered is within close proximity to the image. Any differences between web sites can be accommodated for based on the assisted (e.g. user) nature of the capturing of web page contents. For example, after the initial guess by the javascript, incorrect matches can be altered by the user clicking on the field that was matched incorrectly, and then locating the correct match on the page, and clicking on that. Once the item is submitted, the confidence intervals are updated based on the fields submitted.

[00292] Accordingly, referring to Figure 24, the user device 101D first connects to the gateway 22DD and then requests the desired web page 60D (see Figure 40) that is obtained from the web site 20D. Upon receipt of the java script 95D (or other executable instructions for facilitating the capture of the web page contents), predefined criteria 96D is used to search the rendered page 60D for the desired object(s) (e.g. product including image, title, description,

and price). The criteria 96D is used to compare with the objects located on the web page and if the objects pass the analysis, they are considered as matching candidates for final approval by the user. All candidates are then displayed 97D to the user (see Figure 41). The user can accept 98D the candidate(s) and/or suggest alternative matches (e.g. by clicking on objects displayed on the screen (e.g. different title, price for the correctly identified image and description) based on a visual inspection of the web page displayed on the screen of the device 101D. Acceptance and/or amendment of the candidates can be used to update the pertinent parts of the predefined criteria 96D for subsequent use in matching other candidates. Further, in the case of multiple candidates, the redefined criteria 96D could be used to revise the remaining candidates, before final review by the user.

[00293] It is recognized that the assisted recognition of web page contents could also be used to locate any navigational items that are related to the web page content (e.g. a buy button located adjacent to a product, a bid now button located next to an auction item, etc.).

[00294] Further, this method of web page recognition can be tuned capture the key information on a webpage for different genres of sites. For example, e-commerce websites, news sites, sport etc. The method can capture the product image, title, price & description from a page and then post the information with the URL of the webpage to a server to store the information for the user for later retrieval and use, e.g. a rich bookmark. This allows the user to store rich bookmarks that contain more than just the URL of the website. An example of rich bookmarks 99D lists are shown in Figures 42 and 43, which shows the bookmarks 99D in the context of URL links accessible from a browser menu.

[00295] Example Operation

[00296] Field Attributes

[00297] Image:

[00298] Title:

[00299] Description:

[00300] Price:

[00301] Example

[00302] Site: <http://www.bestbuy.com>

[00303] Link:

<http://www.bestbuy.com/site/olspage.jsp?skuld=7731564&type=product&productCategoryId=pcmcat95100050005&id=1140392418573>

[00304] Source: web page of Figure 40.

[00305] Referring to Figure 42, the following steps can be implemented in user assisted capture of web page contents, namely:

[00306] 1) User navigates to item page

[00307] 2) User clicks FatFreeMobile (activation of desire to connect to gateway 22DD) – Save

[00308] 3) A request is made to fatfreemobile.com (i.e. the gateway 22DD) for the product javascript 95D

[00309] 4) The FatFree server receives the request

[00310] a) The server checks to see if the user is already logged in, if the user is not logged in, the server checks for cookies with the user credentials

[00311] b) The server extracts the requesting site from the referrer section of the http request

[00312] c) The server attempts to the confidence intervals for the site (based on predefined identification criteria 96D).

[00313] d) The server dynamically creates the javascript based on the information from steps (a) and (c).

[00314] e) The server returns the javascript to the client

[00315] 5) The client receives the javascript, which initiates variables required to start the engine, and then launches the engine. Code snippet: `watPM.watStart(window);`

[00316] 6) The function `watPM.watStart(window)` performs the following tasks (e.g. based

on the identification criteria 96D)

[00317] a) Initializes the objects variables

[00318] b) Locates the largest rendered frame

[00319] c) From the largest frame, all <head> and <body> tags are extracted. Code snippet: `getElementsByName('body');`

[00320] d) The remaining tags i.e. <a> <td> Code snippet: `getElementsByName('body');`

[00321] e) A style sheet from FatFreeMobile is then injected into the head of the document

[00322] f) Special characters such as " are replaced with their respective rendered characters i.e. "=

[00323] g) The GUI for FatFreeMobile is injected into the body, as the first element

[00324] i. API call `document.element.insertBefore(new_element);`

[00325] h) Step 0 is then called `setTimeout("top.watPM.watStage(0)", 20);`

[00326] 7) The function `setTimeout("top.watPM.watStage(0)", 20);` performs the following tasks by calling `watScriptX()`

[00327] a) All script tags that are embedded within the page are removed

[00328] i. API call `document.removeElement(element);`

[00329] b) Step 1 is then called `setTimeout("top.watPM.watStage(1)", 10);`

[00330] 8) The function `setTimeout("top.watPM.watStage(1)", 10);` performs the following tasks by calling `watParseIt(0)`. This function looks at all of the tags. However it only process 1000 at a time, for example, to help avoid the warning message a browser prompts with

[00331] "The javascript is not responding". So for each tag the functions performs the following (e.g. based on the identification criteria 96D)

[00332] a) Extract the tag name (i.e. <A>
 <TABLE>)

[00333] b) Ensure the current tag is visible. If the tag is not visible (one of the following styles implies hidden visibility=hidden display=none) the tag is ignored.

[00334] c) The position of the tag (absolute, relative, etc) are extracted from its style property

[00335] d) If the tag is one of the following it is ignored ('LINK','STYLE','HEAD','TITLE')

[00336] i. For example <title>Hewlett-Packard - 42" Plasma HDTV - PL4260N</title> is ignored

[00337] e) If the position (c) is absolute, and the x coordinate < 0 and/or the y coordinate is < 0 the element is ignored

[00338] i. For example <div id="kioskMessage" style="display:none;"> and all of its children are ignored

[00339] f) All javascript actions from the given object are cleared. (i.e. object.onclick will be set to return false;

[00340] i. For example <script language="JavaScript">if(isKiosk){var kioskwarning = document.getElementById("kioskMessage");kioskwarning.style.display = "block";strAdHeight2 = kioskwarning.offsetHeight;}</script> is removed

[00341] g) If the objects tag = IMG or (tag = INPUT and type = image) the object is saved as a candidate for the products image.

[00342] i. For example the product image

[00343] ii. For example not the correct product image, but still an

image.

[00344] h) If the objects tag is in the

following('TD','UL','P','DIV','SPAN','B','H1','H2','H3','H4','H5','H6','STRONG','FONT','BIG') and the objects innerHTML code length is < 1024 (for example) the object is stored as a possible candidate for the products title, price, and description.

[00345] i. For example <td class="Body-Headline" colspan=2>Hewlett-Packard42"

Plasma HDTV
</td> the correct title

[00346] ii. More Options an incorrect title

[00347] iii. <td class="Body">Watch all of your favorite high-definition quality

broadcasts on this 42" plasma TV that features SRS...</td> the correct description

[00348] iv. <td class="Body" valign="top">16:9 widescreen aspect ratio delivers a

cinema-style entertainment experience; 3-2 pulldown for accurate reproduction of film-based sources</td> an incorrect description

[00349] v. <div class="priceblock">Our Price: \$1,199.99
</div> the correct price

[00350] vi. <div class="priceblock">Our Price: \$99.99
</div> an incorrect price

[00351] i) Step 2 is then called setTimeout("top.watPM.watStage(2)", 10);

[00352] 9) The function setTimeout("top.watPM.watStage(2)", 10); performs the following tasks by calling watSetTitles(), which calls watAttrib(hcc,lcc,tcc), (e.g. based on the identification criteria 96D);

[00353] i. var hcc=[2,1]; //initial requirements

[00354] ii. var tcc=[2]; //post location requirements

[00355] iii. var lcc=this.ltitle;

[00356] a) all candidates for titles from step 8 are compared with each other. The top 5 (for example) are selected from the following:

[00357] i. First the objects weight is assigned a numeric value based on their rendered weight. Each objects' weights are compared.

[00358] 1. not defined, normal, and 400 = 400

[00359] 2. bold, bolder and > 400 = 700

[00360] 3. < 400 = 300

[00361] ii. Any ties are broken by the objects rendered size. The size is assigned a numeric value based on its rendered size.

[00362] 1. x pixels = x

[00363] 2. x pt = $4/3 * x$

[00364] 3. HN =

[00365] a. Tag = H1 = 2

[00366] b. Tag = H2 = $3/2$

[00367] c. Tag = H3 = $9/8$

[00368] d. Tag = H4 = 1

[00369] e. Tag = H5 = $13/16$

[00370] f. Tag = H6 = $5/8$

[00371] g. Tag = ELSE = 1

[00372] 4. x % = $x * (16 / 100) * HN$

[00373] 5. x em = $x * 16 * HN$

[00374] 6. xx-small = 10

[00375] 7. x-small = 12

[00376] 8. small = 16

[00377] 9. medium = 18

[00378] 10. large = 24

[00379] 11. x-large = 32

[00380] 12. xx-large = 48

[00381] 13. 1 or -2 = 10

[00382] 14. 2 or -1 = 13

[00383] 15. 3 = 16

[00384] 16. 4 or +1 = 19

[00385] 17. 5 or +2 = 24

[00386] 18. 6 = 32

[00387] 19. 7 = 48

[00388] 20. ELSE = 12

[00389] b) The candidates are then arranged in order based on their distance from the center of the page. The closest to the center would be the first choice. Etc... The center of the page is defined by the confidence intervals

[00390] c) Finally the winning candidate is selected by comparing the confidence interval of the most common winner, the confidence interval of the location, and the weight of each object.

[00391] d) For example, comparing the correct title, and the incorrect title above. Both would evaluate to a weight = 700. The size of the correct item is larger, so it would be ranked ahead. Next the locality of each object would be compared. Since the correct title is closer to the center it would remain ranked higher. The items would then be re-ranked based on their weight. Since their weights are equal the winner is the correct title.

[00392] a. Step 3 is then called `setTimeout("top.watPM.watStage(3)", 10);`

[00393] 10) The function `setTimeout("top.watPM.watStage(3)", 10);` performs the following tasks by calling `watSetDescription()`, which calls `watAttrib(hcc,lcc,tcc)`, (e.g. based on the identification criteria 96D);

[00394] i. `var hcc=[5,-1]; //initial requirements`

[00395] ii. `var tcc=[]; //post location requirements`

[00396] iii. `var lcc=this.ldesc;`

[00397] a) all candidates for titles from step 8 are compared with each other. The top 5 (for example) are selected from the following:

[00398] i. First the objects length of the innerHTML (the length of the source html code the

object contains). The longer the length, the more likely it is a description.

[00399] ii. Second the weight of the object is compared. A detailed explanation was provided in step (9). The -1 signifies that a candidates weight counts as a negative attribute. Therefore, text that is not bold/italic etc is more likely to be a description.

[00400] b) The candidates are then arranged in order based on there distance from the center of the page. The closest to the center would be the first choice. Etc... The center of the page is defined by the confidence intervals

[00401] c) Finally the winning candidate is selected by comparing the confidence interval of the most common winner, the confidence interval of the location.

[00402] d) For example, comparing the correct description, and the incorrect description above. The length of the correct item is larger so it would be ranked ahead. Next the locality of each object would be compared. Since the correct description is closer to the center it would remain ranked higher. The items would then be re-ranked based on their weight, where a stronger weight counts against the item. Since there weights are equal the winner is the correct description.

[00403] e) Step 4 is then called `setTimeout("top.watPM.watStage(4)", 10);`

[00404] 11) The function `setTimeout("top.watPM.watStage(4)", 10);` performs the following tasks by calling `watSetPrice ()`, which calls `watAttrib(hcc,lcc,tcc)`, (e.g. based on the identification criteria 96D);

[00405] i. `var hcc=[6,9,8,2,1]; //initial requirements`

[00406] ii. `tcc=[6,9]; //post location requirements`

[00407] iii. `var lcc=this.ldesc;`

[00408] f) all candidates for titles from step 8 are compared with each other. The top 5 (could change later) are selected from the following:

[00409] iii. First the objects text is searched for a dollar sign (\$). Objects that have a dollar sign will be ranked higher

[00410] iv. Second the objects text is casted to a decimal. If the cast is successful, i.e. the text is a number the element is ranked higher.

[00411] v. Third the objects text is scanned to determine if any numbers exist. If a number is found the object is ranked higher

[00412] vi. Fourth the objects weights are compared. Objects that are bold/italic will rank higher

[00413] vii. Fifth the objects size is compared. The larger the font of the price the more likely it is the products price.

[00414] g) The candidates are then arranged in order based on there distance from the center of the page. The closest to the center would be the first choice. Etc... The center of the page is defined by the confidence intervals

[00415] h) Finally the winning candidate is selected by comparing the confidence interval of the most common winner, the confidence interval of the location, whether or not a \$ sign exists, and whether the text is a numeric.

[00416] i) For example, comparing the correct price, and the incorrect price above. Both would evaluate to true when searching for a dollar sign. Neither item is a decimal, as they both contain text. Both would evaluate to true when searched for numbers. Both weights would evaluate to 700. Finally the size of both items are equal. So the item is essentially tied, and since html is a top down language the first item is ranked higher in our case the incorrect item. Next the locality of each object would be compared. Since the correct price is closer to the center it would now be ranked higher. The items would then be re-ranked based on the dollar sign and decimal tests. Since there both items evaluate to be equal the winner is the correct price.

[00417] j) Step 5 is then called `setTimeout("top.watPM.watStage(5)", 10);`

[00418] 12) The function `setTimeout("top.watPM.watStage(5)", 10);` performs the following tasks by calling `watSetGraphics()`, which calls `watAttrib(hcc,lcc,tcc)`, (e.g. based on the identification criteria 96D);

[00419] a) all candidates for titles from step 8 are compared with each other. The top 5 (could change later) are selected from the following:

[00420] i. First find the rendered width and height of the image.

[00421] ii. Determine the distance from the center of the page

[00422] iii. Compare an object by taking its area – distance to the center. The object that results with the larger number is more likely to be the image.

[00423] iv. For example, comparing the correct image, and the incorrect image above. The area of the correct image is visibly larger than that of the incorrect image. As well the correct image is also visibly closer to the center. Then if the correct image CA, and the incorrect image IA would demonstrate: $\text{area of CA} - \text{distance to middle CA} > \text{area of IA} - \text{distance to center}$. Hence the correct image is chosen.

[00424] b) Step 6 is then called `setTimeout("top.watPM.watStage(6)", 10);`

[00425] 13) The function `watAddItem` takes the guess for image, title, description, and price and displays them to the user, shown in figure 41. The user now has the ability to change a selection by clicking first on the field that was guessed incorrectly. This field will be highlighted in yellow, then locate the correct item on the page, when the correct item is highlighted in yellow, clicking on that item will update the guess.

[00426] 14) The user clicks Save

[00427] 15) A form is posted to FatFreeMobile with the products image, price, title, and description. As well for each field, the x,y location of the field and the guess number is sent to FatFreeMobile

[00428] 16) The server receives the request and updates the database accordingly. The server also downloads the selected image, to help avoid hot linking when displaying products.

[00429] It is recognized that the above assisted capture method can be used as a method to have one or more distributed users help or otherwise be employed to create portions of a signature file one or more distributed users help or otherwise be employed to create portions of a signature file for a web site. For example, a number of users could be assigned different pages from a web site in order to assemble a corresponding signature file for the complete web site, as desired.

[00430] Schema Communication Flow

[00431] The following description provides an example operation of the interaction between the gateway 22DD, the mobile 24D and desktops 26D, and the web pages 60D obtained from the website 20D, based on the requests for content/navigation from the mobile 24D and desktops 26D (see Figure 20).

[00432] Referring to the above figure 45 and Figure 20 with respect to the following steps;

[00433] 1. A client makes a request to the Schema Engine 23D, acting as a proxy 20D, for a specific webpage 60D from a specific domain (e.g. web site 20D);

[00434] 2. The engine receives the request and makes a request to the web site 20D for the specified page and retrieves the web page code into memory. This may not include objects on the page such as pictures that are inserted at the time of rendering;

[00435] 3. The engine in parallel makes a request to the signature repository to acquire the signature file 64D for the domain using the domain in the URL as the key to retrieve the signature file, for example;

[00436] 4. The engine does not render the page but instead uses the code in the signature file as instructions to extract the desired data from the web page, such that the desired data is defined for a particular request type received by the gateway from the mobile 24D/desktop 26D and/or for a predefined mobile 24/ desktop 26 platform (e.g. having knowledge of device display capabilities – screen size, resolution, and other parameters useful in determining the way in which the data is capable of being displayed on the device 101D);

[00437] 5. The data can optionally be stored in a local data repository;

[00438] 6. The engine transmits the data to the client that requested the page; and

[00439] 7. The client could be a browser application that displays the data or could be an application that renders the data (e.g. see navigational menu 300D example described below).

[00440] Further below is described a section on a detailed explanation on how the Schema Engine understands the signature file syntax and processes a webpage, as proxied between the mobile/desktop and the web site.

[00441] It is noted that an example embodiment of the engine 23D and the signature file 64D used to interpret the web page 60D and subsequently send revised/reformatted web page content/navigation data to the screen with limited real estate requirements (e.g. mobile) is provided in Appendix A.

[00442] Referring to Figures 20 and 23, shown is an example operation of steps implemented to obtain and satisfy a web page request from the user device 101D (e.g. mobile) by the gateway 22DD. Step 214 - clients makes a request for the ABC ComTech Corp. page shown in figure 26 and after steps 1, 2 and 3 above take place, the Schema Engine has the

ABC ComTech Corp. webpage code and signature file loaded. The remaining steps are a detailed description of the above step 4, where the engine does not render the page but instead uses the code in the signature file as instructions to extract the desired data from the web page. Accordingly, step 201 - Schema Engine confirms that input HTML is from ABC ComTech Corp..ca and this signature file is that of ABC ComTech Corp..ca, step 202 – Schema Engine sets a global variable to append "&test%5Fcookie=1" to all requests and sets main index to http://www.ABC ComTech Corp..ca/home.asp?newlang=EN&logon=&langid=EN, step 203 - Schema Engine then tries to determine the page type by checking existence of string identifiers for each page family, step 204 - Schema Engine then jumps to the "item_elements" section of the signature file that contains instructions for extracting the object elements for the page, step 205 - Schema Engine trims HTML scope, step 206 - Schema Engine extract image, step 207 - Schema Engine extracts title, step 208 - Schema Engine extracts price, step 209 - Schema Engine extracts sale price, step 210 - Schema Engine extracts description, and step 211 - Schema Engine assembles and returns all extracted data for display on the mobile, dependent on the format as predefined suitable for the mobile display capabilities (e.g. screen size, resolution, etc.). At step 212, the engine transmits the data to the client that requested the page (as per step 6 above).

[00443] It is recognized that the above described steps 206-210 can be for the extraction of web It is recognized that the above described steps 206-210 can be for the extraction of web example of web page content/navigation items that are obtained by the engine from the web page 60, using the signature file as a guide for the extraction. It is recognized that the engine can also have a series of formatting rules, not shown, for use with the extracted data in generating a page with the extracted data that is suitable for display on the target device 101D (e.g. desktop, mobile). It is recognized that the formatting rules can be system and/or user defined and can include such parameters such as but not limited to: object positioning, object colour, object size, object shape, object font/image characteristics, background style, and navigational item display (e.g. in menu 300D or embedded along with the content in the generated page for display on the target device).

[00444] User Interface Optimization by separating web page content and navigation

[00445] Schema Solution – Gateway 22DD

[00446] Although a Schema Engine 23D of the gateway can automatically determine

whether to send back menus or content for a given web page, an Schema client (e.g. mobile 24D and/or desktop 26D) has the ability to explicitly request either the navigation menu or the content for a page and the Schema Engine provides each output accordingly. On each screen of the user device 101D (e.g. mobile or desktop) the user either sees the navigation menus or the page content for the respective page. This method is accomplished using the Schema Engine and signature file, further described below. Figure 26 shows the output the client receives from the Schema Engine, when it extracts the navigation from a typical web page (that has both navigation and content) The Schema engine takes in ABC ComTech Corp. page marked "1" and outputs page marked "2" to the schema client. Figure 27 shows the output the client receives from the Schema Engine, when it is extracts the content (list items) from a typical webpage. The Schema Engine takes in ABC ComTech Corp. page marked "3" and outputs page marked "4" to the schema client.

[00447] This demonstrates how the navigation and content from web pages can effectively be separated and transmitted by the Schema Engine to the schema client.

[00448] Packaging page content and navigation menu data into a mobile application

[00449] Page Content

[00450] From the above example it is apparent that the Schema Engine is able to output navigation and content data independently as the result of a given web page input. Packaging content into a mobile application (e.g. application hosted by the mobile device or desktop device) entails rendering the data output of the Schema Engine in a client mobile application instead of the web browser. In the current embodiment, as an example, a web browser makes a request for a page and receives the content data as the response that it renders. The mobile application can similarly make a request for the page, as the browser could, and render the data received from the Schema Engine. See Figure 28 as an example of the rendering of a mobile client application.

[00451] To fully package a website into an application, the navigation functions of the website (browsing) and special features (buy item, check availability, and other buttons/links) can be inserted into a menu 300D of the application, see Figure 29. One advantage of doing so is that the user can invoke the application menu for navigation rather than loading a new page or screen refresh to view the navigational items displayed in the web page. In another embodiment, navigation options can also be inserted in pop-up menus and dialogue boxes of

the applications using a similar approach as described below. It is recognized that the gateway can be used as a data flowthrough mechanism, with respect to data on navigational items extracted from the webpage 60D. Accordingly, the navigational item data would be tagged by the gateway for subsequent interpretation by the client application (e.g. mobile), such that the client application would recognize the navigational item data for insertion into the navigation menus 300D, insertion as embedded into the published content displayed on the screen, or a combination thereof. It would be up to the mobile application, for example, to determine the manner in which the navigational item data is to be used in conjunction with the publish content data for any respective web page, as desired.

[00452] The pages marked “2” and “4” in figures 26 and 27 respectively show how the navigational features and content features are divided into 2 separate pages/files. A user has to switch pages to toggle between navigation and content.

[00453] Figure 28 shows an example client application’s content area. Note that the navigational features of the web page can be selected through the menu 300D that is linked to the actual navigational instructions of the respective Web page(s), which is invoked in a single click (Figure 23) that has the website navigation options displayed to the used as an application menu 300D rather than as navigational features displayed on the web page.

[00454] Navigation menus 300D

[00455] A menu item can be statically created at compile time and its function is known at compile time for web pages. The Schema Engine can dynamically create menu items at run time. Assume that a navigational item is meant to be processed on the client that accomplishes inserting menu items dynamically. If the mobile application was passed the extracted navigational information from the engine, the application would insert the items into the application via MenuItem(name, URL), for example. In this case, it is the engine that would pass the data (name, URL) that indicates the data as a potential menu name and corresponding URL as parameters. The application would insert the data as a menu item into the application menu 300D, such that these parameters would then be linked to the corresponding respective menu item selection. The method described above dynamically inserts navigational items of the web page into the application menu of the application used to interact with the web site contents. The implementation of this method can differ depending on the application and platform of the device 101D. It should be recognized that the menu items are related to navigation of the content in the web pages rather than only between the web

pages themselves.

[00456] The following steps outline an example process for dynamically inserting menu items into a mobile application menu:

[00457] 1. The client application makes a request for the navigation items of a web page

[00458] 2. The Schema Engine receives the web page (marked “1” in Figure 26)

[00459] 3. The Schema Engine extracts the navigational items and sends the data set (menu name, URL) to the client. Table I shows the output for the first 5 navigational items from the web page of Figure 26.

[00460] 4. The client receives the navigational items and calls a createMenuItem() method with each [menu name, URL] set received, thus displaying the navigational items as menu items. It is recognized that the menu items can be displayed overtop of the content displayed on the screen of the device 101D, where the navigational items are no longer displayed adjacent to the content (as formatted in the original web page) and rather assembled/combined and displayed in a separate navigational menu for navigating the content of the web site.

[00461] At this point, the navigation items for the page are loaded into the application menu 300D. A user can click a menu item, which will result in the application invoking the URL associated with the menu name and thus facilitating the display of the web site content associated with the menu item (representing the original navigational item). For example, the menu item can be used to invoke one of the navigational items of the web site (e.g. “buy item”), rather than just navigate between pages.

[00462] Using this method both content and navigation features can be simultaneously retrieved for a given page. For example in the diagram if the user selects the navigational name “computers” the URL page request will be sent to the Schema Engine that will respond to the client with the content for that page as well as the navigation items (in menu format for example) for that page.

[00463] The content is rendered in the application as previously described and the web page navigational items are inserted into the application menu as described above. Accordingly, the contents of the navigational menu 300D for any particular web page is

dependent upon the navigational items that are contained or are otherwise associated with that web page as configured by the web site. In this case, both traditional content 50D and the navigational features 54D can be treated as components for each of the web pages. Hence, the web pages of the web site (through use of the signature file described below) can be represented as having web page contents that includes both the content 50D and the navigational items 54D. In this sense, each menu 300D for each page is dynamically created based on the navigational items resident/associated with that page (and page content 50D).

[00464] Further, it is recognized that some navigational items 54D can remain on the web page as displayed (e.g. embedded with the displayed content 50D), can be represented as separate menu 300D items, or a combination thereof.

[00465] Maintaining a Transactional Session across Devices 101D

[00466] Referring to Figure 25, in general the state of a user's browsing session can have a number of different perspectives, such as but not limited to: 1) a Browser/application 207D perspective including navigation history 82D across the current browsing session, such that the browser/application 207D can keep track of the address (e.g. URL) of pages previously visited (back and forward buttons) as well as the current page the user is on. The current page can also be captured through a bookmark., as is known in the art; and 2) a web site perspective such that session information relating to the specific website can include website state information 80D like a session ID, site preferences, and shopping cart items, for example. This website state information (particular to the user's interaction with the web site – e.g. tracked through the users' ID such as login ID or device 101D ID) can either be stored on the client (user's machine 101D) in a browsing history file (e.g. a cookie) or on the server device 101D associated with a user's session ID (e.g. website 20D and/or the gateway 22DD). In the case of the gateway 22DD, a table/memory 92D can be used to store or otherwise monitor the history 82D and/or the state information 80D with respect to each user transaction that is in a pending/unfinished state (e.g. user has browsed/interacted with a number of web pages to a certain stage for product purchase, but has not yet progressed to the point of transaction completion – e.g. confirmation of payment and shipping information of a selected product).

[00467] It is noted that a cookie can be referred to as a small text file of information 80D that certain Web sites can attach to a user's hard drive (of their device 101D) while the user is browsing the Web site. The Cookie can contain information such as user ID, user preferences, archive shopping cart information, etc. Since the web sites can be inherently stateless, these

cookies or other session history equivalents can also be a good way to create and maintain state from a website's perspective, as implemented by the environment 10D as further described below. Further, a bookmark can be referred to as a process of saving a URL (e.g. network 11D address) in the web browser/application 207D. The bookmark 82D allows the user to return to a particular web site or web page by making a record of the corresponding network address. A bookmark however may not capture the state (data entered/requested in the process of transaction completion) of a user's browsing session, rather the bookmark serves as a reference point for the location of the web page/web site last visited by the user. One can appreciate that a bookmark captures may only a fragment of a user's browsing session, for example only the address of current page that the user was on.

[00468] Accordingly, saving and restoring a user's session can have one or more different components, such as but not limited to: saving and restoring the current page and navigation history 82D; and/or saving and restoring the specific website's transactional state 80D pertaining to the user (e.g. using the respective cookie for the transaction).

[00469] Saving a user's browsing session

[00470] Saving navigation history can be accomplished by saving the current page (saving the URL such as a bookmark would do) and optionally gathering the browser's navigation history. For example on a mobile client, all pages that a user requests can be saved on the client or on a remote server.

[00471] For example, when a client browser or application (mobile or desktop) makes an http request, a request comes back including 2 parts, an http header and the http content. One of the instructions in the http header is a "set cookie" command. A browser or client application uses that command to create and maintain the cookie on the client. When the browser or client application makes a web page request, it can pass all the cookies back to the website to maintain state. Because cookie information can be in plain text in a header, it can readily be extracted by a mobile client application. One embodiment of the browser/application 207D is to collect cookies on the desktop/mobile is to use a browser plug-in or state application 88 to retrieve cookies from the "temporary internet folder" of the device 101D where cookies are typically stored and transmit them to the remote server or database. Saving cookies is a way to save the user's state from the website perspective.

[00472] Accordingly, the user's transaction can be saved through use of the history 82D

and/or information 80D. For example, if the user wishes to save a particular transaction-in-progress, the user can notify the gateway 22DD of the intension and the gateway can save the history 82D, information 80D in the memory 92D, for later use in reactivating the particular transaction-inprogress. It is recognized that the data captured as a rich bookmark could also be used in the data 80D,82D as desired.

[00473] Restoring a user's browsing session

[00474] Restoring the current page can be accomplished by making a request by the user for the current page on the client application/browser (mobile or desktop). The gateway is then responsible for sending to the current device 101D (either the same or different device by which the transaction was last done with) a transaction continuance package 84D that is related to the saved particular transaction-in-progress from the memory 92D, which would contain data such as but not limited to: the saved navigation history for use in populating the navigation history of the user device; and/or all saved cookies for use in restoring website state information by placing the cookies into the appropriate location that the browser or client application uses to create and manage cookies, e.g. the "temporary internet folder".

[00475] One aspect is that the application 88D could synchronize all cookies from the desktop to the mobile device or vice versa. This way, user preferences for all web sites (including remembered login ids, for example) could be always synchronized between a mobile device and the desktop. Further, it is recognized that the memory 92D could be used to remember the device on which the transaction-in-progress was last implemented on and to therefore try to maintain the formatting of web pages 86D as displayed previously for the user activity with respect to the transaction-in-progress. One example of this is to keep the simplified formatting of the web pages done for the mobile display the same for display of similar pages on the desktop, even though sufficient desktop screen space is available to display the original content and format of the web pages. Similarly, for transactions started on the desktop, the continuance of the web pages on the mobile, with respect to desktop formatted webpages, could be retained (e.g. through re-organization of the pages and wrap content around the screen, or used of the WAP standard to spatially divide a page (usually vertically) into a number of pages and allow the user to navigate between each page section to view a page). The maintaining of the look and feel of the particular web page content could be useful in keeping the user from becoming confused between format changes of the web pages. Further, for example the user could select a certain web page format for display through the gateway (e.g. original or otherwise simplified format), in the event that the user

anticipates changing devices (e.g. desktop to mobile) to continue and complete the transaction, as desired.

[00476] Another extension of the concept of saving the transaction-in-progress is that variables such as an affiliate revenue sharing code can be included in the URL. That way, the user can start browsing from a PC or mobile device and save their session based on the code. When they restore the session on another PC or mobile device, the revenue share would be received by appropriate entity based on the code usage.

[00477] Caching

[00478] There can be cache points on the engine as well as the client. The cache can consist of the actual webpage or the data output of the webpage. Cache's can be build upon request or output can be pre-cached to optimize the user experience. A combination of the above on different kinds of pages can be used to develop caching schemes for usage. Another aspect of the engine is that it can be used to crawl an entire website with the corresponding signature file and build a complete database of product information from the website automatically.

[00479] Pre-Caching (Offline Synchronization) of website content to a mobile client

[00480] Another aspect is the ability for a user to load website content (pre-caching) from the Schema Engine to the client in larger segments instead of page by page. This could either be done through an application on an internet enabled PC when the mobile device is connected to the PC or directly from the mobile device when the user has a wireless data connection available. Once the desired content is on the mobile device, the user could browse the content without a wireless connection.

[00481] In the current examples, when a user selects a menu item from a menu page, for example "computers" in figure 35, the Schema Engine fetches the corresponding page to the menu item which could be a sub-menu item or a list page. Another embodiment of the invention would allow the user to select the category "computers" in figure 35 and the Schema Engine would automatically traverse all sub menus in the category and cache all information including sub menus, list pages and item pages in the category. Other examples are sections of a news site including "headlines", "business", "sports" etc. Using this method a user could select categories that could be pre-cached on the device for offline browsing at a later time.

The same techniques could be applied to cache an entire website for data mining or searching (for example a price comparison website that wished to have indexes of multiple e-commerce sites). Further, it is recognized that no network calls could be required in the event of precaching. For example, precache could happen at the beginning of day via the user desktop and then synched to the mobile via a wired connection (for example), so that user can surf precached information offline. For example, using signature file to grab content based on precaching criteria, this could be used to generate the precache database. Examples of wanting to build a local precached database on the device could include: 1) for browsing situations for no/interrupted connection potential; or 2) for fast browsing. Further, for price comparison websites, they can crawl the web and build a comparison pricing information to make available to the public or other subscribers.

[00482] Appendix A

[00483] Platform Overview

[00484] Referring to figure 45, the following steps can be effected:

[00485] 1. A client (1) makes a request to the Schema Engine(4), acting as a proxy, for a specific webpage (2) from a specific domain

[00486] 2. The engine (4) receives the web page code (2) into memory. This typically does not include objects on the page such as pictures that are inserted at the time of

[00487] rendering.

[00488] 3. The engine (4) in parallel makes a request to the signature repository (3) to acquire the signature file for the domain using the domain in the URL as the key to retrieve the signature file

[00489] 4. The engine does not render the page but instead uses the code in the signature file as instructions to extract the desired data from the web page (6).

[00490] 5. The data can optionally be stored in a data repository (5)

[00491] 6. The engine (4) transmits the data to the client (1) that requested the page

[00492] 7. The client (1) could be a browser application that displays the data or could be an

application that renders the data

[00493] Schema Engine Detailed Walk Through

[00494] Assume that a clients makes a request for the ABC ComTech Corp. page shown in figure 26. After steps 1, 2 and 3 take place, the Schema Engine has the ABC ComTech Corp. webpage code and signature file loaded as shown below. The next few pages are a detailed explanation of step (4)

[00495] Code Snippet of ABC ComTech Corp. page shown in Figure 26

[00496] <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

[00497] "http://www.w3.org/TR/html4/loose.dtd">

[00498] <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />

[00499] <script language="JavaScript" src="/javaScript/productdetail.js"></script>

[00500] <html>

[00501] <head>

[00502] <meta NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">

[00503] <link rel="SHORTCUT ICON" href="http://www.ABC ComTech Corp..ca/favicon.ico">

[00504] <title >ABC ComTech Corp.: Computers: Laptops: Acer Aspire AMD Turion 64 X2

[00505] Dual Core TL-52 1.60GHz Laptop (AS9300-5383F) - French - FS Exclusive</title>

[00506] </head>

[00507] <body onLoad="onLoadAction(); return false;" onResize="setDDOnResize(); return false;" leftmargin="10" rightmargin="10" topmargin="10" bottommargin="10">

[00508] <!--=====

[00509] Server=fsweb84

[00510] =====//-->

[00511] <script language="JavaScript" src="/javaScript/search.js"></script>

[00512] <table align="center" cellpadding="0" cellspacing="0" class="table-outer-width">

[00513] <tr>

[00514] <td>

[00515] <table width="100%" border="0" cellpadding="0" cellspacing="0">

[00516] <tr valign="top">

[00517] <td colspan="3" class="bg-header-border"></td>

[00518] </tr>

[00519] <tr>

[00520] <td class="bg-header-border"></td>

[00521] ...

[00522] Code Snippet of Schema ABC ComTech Corp. Signature file retrieved by Schema Engine

[00523] <?xml version="1.0" encoding="ISO-8859-1" ?>

[00524] <site>

[00525] <version major="1" minor="2"/>

[00526] <url location="http://www.ABC ComTech Corp..ca" key="ABC ComTech Corp..ca" name="ABC ComTech Corp " />

[00527] <advanced>

[00528] <append_link value="&test%5Fcookie=1" />

[00529] <index_link

[00530] value="http://www.ABC ComTech
Corp..ca/home.asp?newlang=EN&logon=&langid=EN"

[00531] />

[00532] </advanced>

[00533] <page_type>

[00534] <lookup type="pex" action="locate_string" name="list_elements"

[00535] id="mylist_1" ref="Sort or compare products" alt1="Sort products" />

[00536] <lookup type="pex" action="locate_string" name="item_elements"

[00537] id="myitem_1" ref=""product-details-prd-title"" />

[00538] <lookup type="pex" action="locate_string" name="menu_elements"

[00539] id="mymenu_2" ref="anc-lhsnav-subItem" />

[00540] <lookup type="pex" action="locate_string" name="menu_elements"

[00541] id="mymenu_1" ref="product-table" />

[00542] <lookup type="pex" action="locate_string" name="item_elements"

[00543] id="myitem_1" ref="*" />

[00544] </page_type>

[00545] <list_elements id="mylist_1">

[00546] <paging>

[00547] <page_variable value="page" />

[00548] <page_start value="0" />

[00549] <lookup type="pex" action="get_string" name="link" ref="Next
location="before" start="<a class=" end=""include_sz="1" strip_tags="1" />

[00550] </paging>

[00551] <actions>

[00552] <lookup type="pex" action="move_ptr" ref="Sort or compare

[00553] products" alt1="Sort products" />

[00554] ...

[00555] Step 1: Schema Engine confirms that input HTML is from ABC ComTech Corp..ca and this signature file is that of ABC ComTech Corp..ca

[00556] <?xml version="1.0" encoding="ISO-8859-1" ?>

[00557] <site>

[00558] <version major="1" minor="2"/>

[00559] <url location="http://www.ABC ComTech Corp..ca" key="ABC ComTech Corp..ca" name="ABC ComTech Corp. " />

[00560] Step 2: Schema Engine sets a required global variable to append "&test%5Fcookie=1" to all requests and sets main index to http://www.ABC ComTech Corp..ca/home.asp?newlang=EN&logon=&langid=EN

[00561] <advanced>

[00562] <append_link value="&test%5Fcookie=1" />

[00563] <index_link

[00564] value="http://www.ABC ComTech Corp..ca/home.asp?newlang=EN&logon=&langid=EN"

[00565] />

[00566] </advanced>

[00567] Step 3: Schema Engine then tries to determine the page type by checking existence of string identifiers for each page family

[00568] <page_type>

[00569] <lookup type="pex" action="locate_string" name="list_elements" id="mylist_1"
ref="Sort or compare products" alt1="Sort products" />

[00570] <lookup type="pex" action="locate_string" name="item_elements" id="myitem_1"
ref=""product-details-prd-title"" />

[00571] <lookup type="pex" action="locate_string" name="menu_elements"
id="mymenu_2" ref="anc-lhsnav-subItem" />

[00572] <lookup type="pex" action="locate_string" name="menu_elements"
id="mymenu_1" ref="product-table" />

[00573] <lookup type="pex" action="locate_string" name="item_elements" id="myitem_1"
ref="*" />

[00574] </page_type>

[00575] ABC ComTech Corp. web page code snippet:

[00576] ...

[00577] <td valign="top" width="430">

[00578] <table width="100%" border="0" cellpadding="0" cellspacing="0"

[00579] class="product-details-table">

[00580] <tr><td></td><td align="right"><a

[00583] href="http://www.ABCComTechCorp.ca/informationcentre/EN/content_feedback.as
p?sku_id=0665000FS10086374&title=Acer+Aspire+AMD+Turion+64+X2+Dual+Core+TL%2D
52+1%2E60GHz+Laptop+%28AS9300%2D5383F%29+%2D+French+%2D+FS+Exclusive&m
an=Acer&logon=&langid=EN">Feedback</td></tr>

[00584] <tr>

[00585] <td colspan="2" class="product-details-price"></td>

[00587] <td class="product-details-price-trc"></td>

[00588] </tr>

[00589] <tr>

[00590] <td colspan="2" class="product-details-prd-title">Acer Aspire AMD Turion 64 X2 Dual Core TL-52 1.60GHz Laptop (AS9300-5383F) - French - FS Exclusive</td>

[00591] <td class="product-details-r-bdr"> </td>

[00592] </tr>

[00593] ...

[00594] Schema Engine does not find "Sort or compare products" or "Sort products" in the web page so this page is not from the List family. The engine continues to check the next string.

[00595] Schema Engine finds ""product-details-prd-title"" and identifies the page as part of the Item family (item_elements).

[00596] Step 4: Schema Engine then jumps to the "item_elements" section of the signature file that contains instructions for extracting the object elements for the page

[00597] <item_elements id="myitem_1">

[00598] <actions>

[00599] <lookup type="pex" action="move_ptr" ref="</head>" />

[00600] </actions>

[00601] <element>

[00602] <lookup type="pex" action="get_string" name="image" ref="largeimageref" location="after" start="

[00603] <lookup type="pex" action="get_string" name="title" ref="productdetails- prd-title" location="after" start="<span" end="" include_sz="1" strip_tags="1" />

[00604] <lookup type="pex" action="get_string" name="price" ref="our price:" location="after" start="<td" end="</td>" include_sz="1" strip_tags="1" />

[00605] <lookup type="pex" action="get_string" name="sale_price" ref="sale price:" location="after" start="<td" end="</td>" include_sz="1" strip_tags="1" tolerance="1" />

[00606] <lookup type="pex" action="get_string" name="description" ref="detailbox-text" location="middle" start="<p" end="</p>" include_sz="1" strip_tags="1" />

[00607] </element>

[00608] </item_elements>

[00609] Step 5: Schema Engine trims HTML scope

[00610] <actions>

[00611] <lookup type="pex" action="move_ptr" ref="</head>" />

[00612] </actions>

[00613] ABC ComTech Corp. web page code snippet:

[00614] <html>

[00615] <title>

[00616] ...

[00617] </head>

[00618] <body onLoad="onLoadAction(); return false;" onResize="setDDOnResize(); return

[00619] false;" leftmargin="10" rightmargin="10" topmargin="10" bottommargin="10">

[00620] ...

[00621] </html>

[00622] The engine discards all code before "</head>" setting the upper limit .

[00623] Step 6: Schema Engine extract image

[00624] <element>

[00625] <lookup type="pex" action="get_string" name="image"

[00626] ref="largeimageref" location="after" start="" />

[00627] ABC ComTech Corp. web page code snippet:

[00628] <DIV id="largeImageRef" style="display:none">

[00629] <a href="#"

[00630] onClick="openWindowAdv('http://www.ABC ComTech Corp..ca/popup/largeimagepopup.asp?logon=&langid=EN&title=Acer+Aspire+AMD+Turion+64+X2+Dual+Core+TL%2D52+1%2E60GHz+Laptop+%28AS9300%550, 524, 0, 0, 0, 0, 0, 0);' title="Click here for larger view">" that appears after next appearance of "largeimageref". The string returned is the path to the product image.

[00632] Step 7: Schema Engine extracts title

[00633] <lookup type="pex" action="get_string" name="title" ref="product-details-prd-title" location="after" start="<span" end="" include_sz="1" strip_tags="1" />

[00634] ABC ComTech Corp. web page code snippet:

[00635] <tr>

[00636] <td colspan="2" class="product-details-prd-title"><span class="tx-heading3-

dgrey">Acer Aspire AMD Turion 64 X2 Dual Core TL-52 1.60GHz Laptop (AS9300-5383F) -
French - FS Exclusive</td>

[00637] <td class="product-details-r-bdr"> </td>

[00638] </tr>

[00639] Then Schema Engine returns the string in between first "<span" and "" including first and last element that appears after next appearance of "product-detailsprd-title", excluding any mark up language. The string returned is the title.

[00640] Step 8: Schema Engine extracts price

[00641] <lookup type="pex" action="get_string" name="price" ref="our price:"
location="after"

[00642] start="<td" end="</td>" include_sz="1" strip_tags="1" />

[00643] ABC ComTech Corp. web page code snippet:

[00644] <tr>

[00645] <td class="tx-strong-grey">Our price:</td>

[00646] <td class="tx-normal-grey" style="text-align: right">\$1,059.99</td>

[00647] </tr>

[00648] Then Schema Engine returns the string in between first "<td" and "</td>" including first and last element that appears after next appearance of "our price:", excluding any mark up language. The string returned is the price.

[00649] Step 9: Schema Engine extracts sale price

[00650] <lookup type="pex" action="get_string" name="sale_price" ref="sale price:"
location="after" start="<td" end="</td>" include_sz="1" strip_tags="1" tolerance="1" />

[00651] ABC ComTech Corp. web page code snippet:

[00652] <tr>

[00653] <td class="tx-strong-red">Sale Price:</td>

[00654] <td class="tx-normal-red" style="text-align: right">\$999.99</td>

[00655] </tr>

[00656] Then Schema Engine returns the string in between first "<td" and "</td>" including first and last element that appears after next appearance of "sale price:", excluding any mark up language The string returned is the sale price.

[00657] Step 10: Schema Engine extracts description

[00658] <lookup type="pex" action="get_string" name="description" ref="detailbox-text" location="middle" start="<p" end="</p>" include_sz="1" strip_tags="1" />

[00659] </element>

[00660] </item_elements>

[00661] HTML:

[00662] <p class="detailbox-text">

[00663] Decked out with an impressive 17" Acer CrystalBrite widescreen display, the Aspire 9300 enhances multitasking productivity and gaming pleasure. More Info

[00664] </p>

[00665] Then Schema Engine returns the string in the middle of "<p" and "</p>" including first and last element on the occurrence of "detailbox-text ", excluding any mark up language. The string returned is the description.

[00666] Step 11: Schema Engine assembles and returns all extracted data

Image	/multimedia/products/regular/10086374.gif
Title	Title Acer Aspire AMD Turion 64 X2 Dual Core TL-52 1.60GHz Laptop (AS9300-5383F) - French - FS Exclusive

Price	\$1,059.99
Sale Price	\$999.99
Description	Description Decked out with an impressive 17" Acer CrystalBrite widescreen display, the Aspire 9300 enhances multitasking productivity and gaming pleasure.

[00667] Signature File

[00668] Example Source (*.ffs)

Genre	Title	File
e-Commerce	ABC ComTech Corp..ca.ffs	

[00669] ABC ComTech Corp. Page Family Signature Explanation

[00670] 1 <page_type>

[00671] 2 <lookup type="pex" action="locate_string" name="list_elements" id="mylist_1" ref="Sort or compare products" ref_alt_1="Sort products" />

[00672] 3 <lookup type="pex" action="locate_string" name="item_elements" id="myitem_1" ref=""product-detailsprd- title"" />

[00673] 4 <lookup type="pex" action="locate_string" name="menu_elements" id="mymenu_2" ref="anc-lhsnav-subItem"

[00674] />

[00675] 5 <lookup type="pex" action="locate_string" name="menu_elements" id="mymenu_1" ref="product-table" />

[00676] 6 <lookup type="pex" action="locate_string" name="item_elements" id="myitem_1" ref="*" />

[00677] 7 </page_type>

[00678] The Schema Engine processes the <page type> tag by registering the identification strings for each page family. By doing that, when a webpage is sent to the engine as input, the

[00679] engine is able to identify the page family by its unique string.

[00680] action="locate_string"

[00681] command to check for the existence of a string

[00682] name="

[00683] identifies the type of page family for each identified family

[00684] id="

[00685] assigns an id to the page family that is used across the signature file

[00686] When the Schema Engine is passed a web page and the signature file, the first step is to identify the page type which then instructs the engine to the corresponding list_elements tag for the page family.

[00687] ABC ComTech Corp. List Family Signature Explanation

[00688] 1 <list_elements id="mylist_1">

[00689] 2 <paging>

[00690] 3 <page_variable value="page" />

[00691] 4 <page_start value="0" />

[00692] 5 <lookup type="pex" action="get_string" name="link" ref="Next " location="before" start="<a class=" end="" include_sz="1" strip_tags="1" />

[00693] 6 </paging>

[00694] 7 <actions>

[00695] 8 <lookup type="pex" action="move_ptr" ref="Sort or compare products" ref_alt_1="Sort products" />

[00696] 9 </actions>

[00697] 10 <element>

[00698] 11 <lookup type="pex" action="get_string" name="link" ref="thumbnail" location="before" start="" />

[00699] 12 <lookup type="pex" action="get_string" name="image" ref="thumbnail" location="middle" start=""" end=""" />

[00700] 13 <lookup type="pex" action="get_string" name="title" ref="class="tx-strong-dgrey"" location="after" start="<a href=" end="" include_sz="1" strip_tags="1" />

[00701] 14 <lookup type="pex" action="get_string" name="price" ref="pricepill/" location="after" start="/" repeat_start="1" end=".gif" tolerance="1" />

[00702] 15 <lookup type="pex" action="move_ptr" ref="pricepill/" />

[00703] 16 </element>

[00704] 17 </list_elements>

[00705] Once the engine has identified that the page is of the "mylist_1" family the engine finds the spots in the signature file that contains the signature for the objects and elements of the family.

[00706] <paging>...</paging>

[00707] Contains paging attributes of the mylist_1 family. The tags contain instructions to find the number of pages on the list page and generates the links for each of the page links

[00708] <actions>...</actions>

[00709] The action tag instructs the engine to move the scan pointer to the section on the page right before the main list content of the page. This allows the engine to only scan the relevant area, discarding all the code preceding it. This can be important because it can eliminate ambiguity and repetition by instructing the engine on precisely which parts of the page to scan

[00710] <elements>...</element>

[00711] Explanation of the lookup command:

[00712] Lookup type="pex": string lookup

[00713] Action ="get_string": this action type actually return a value back that is the desired element of the object.

[00714] Name="link": the object element, in this case the link to the product page

[00715] ref="thumbnail": the reference string that identifies where to find the value of the link

[00716] location="before": The value of the link is before the ref string

[00717] start="<a href="": look for the ref string after this value

[00718] end="">": look for the ref string before this value

[00719] Line 11 for example instructs the engine to look for a reference of the string "thumbnail", then locate the value between the start and end strings specified to the left of reference point. The element, which is the link to the product page in this case, is before the reference string and its value is to be extracted and returned.

[00720] The last lookup with action="move_ptr" in the element tag instructs the engine to move the pointer past the first object to get ready to repeat the instructions to scan in the element of the second object on the list page.

[00721] Note: If you attach "advance_ptr" to a lookup, this will also advance the pointer (this can be used if ordering in list page exists)

ABC ComTech Corp. Search Family Signature Explanation

[00722] 1 <search_elements id="mysearch_1">

[00723] 2 <settings>

[00724] 3 <search_path value="http://www.ABC ComTech Corp..ca/search/searchresult.asp?logon=&langid=EN&search=KWS" />

[00725] 4 <search_variable value="keyword" />

[00726] 5 </settings>

[00727] 6 <paging>

[00728] 7 <page_variable value="page" />

[00729] 8 <page_start value="0" />

[00730] 9 <lookup type="pex" action="get_string" name="link" ref="Next " location="before" start="<a href=" repeat_start="1" end="" include_sz="1" strip_tags="1" />

[00731] 10 </paging>

[00732] 11 <actions>

[00733] 12 <lookup type="pex" action="move_ptr" ref="bg-compare-hero" />

[00734] 13 </actions>

[00735] 14 <element>

[00736] 15 <lookup type="pex" action="get_string" name="link" ref=">" location="after" start="" />

[00737] 16 <lookup type="pex" action="get_string" name="image" ref="<a href" location="after" start="

[00738] 17 <lookup type="pex" action="get_string" name="title" ref="class="tx-strong-dgrey"" location="after" start="<a href=" end="" include_sz="1" strip_tags="1" />

[00739] 18 <lookup type="pex" action="move_ptr" ref="bg-compare-hero" />

[00740] 19 </element>

[00741] 20 </search_elements>

[00742] Once the engine has identified that the page is of the "mysearch_1" family the engine finds the spots in the signature file that contains the signature for the objects and elements of the family, shown above.

[00743] <settings>...</settings>

[00744] Contains any page specific manual overrides such as excluding certain menu items or and customization, modification of a menu that may need to be done In this example, value of form variable "keyword" will be posted to "http://www.ABC ComTech Corp..ca/search/searchresult.asp?logon=&langid=EN&search=KWS"

[00745] <paging>...</paging>

[00746] Manages paging for the search pages

[00747] <actions>...</actions>

[00748] Instruct the engine to move the scan pointer to the string "bg-compare-hero" and start looking for elements from there

[00749] <element>...</element>

[00750] Contains lookup instructions for each object element as previously described.

[00751] ABC ComTech Corp. Menu Family Signature Explanation

[00752] 1 <menu_elements id="mymenu_1">

[00753] 2 <settings>

[00754] 3 <black_list value="Site Index##ReClaim™ Insurance Replacement" />

[00755] 4 </settings>

[00756] 5 <actions>

[00757] 6 <lookup type="pex" action="move_ptr" ref="bg-lhsnav-title" />

[00758] 7 <lookup type="pex" action="end_ptr" ref="</table>" />

[00759] 8 </actions>

[00760] 9 <element>

[00761] 10 <lookup type="pex" action="get_string" name="link" ref=""

location="after" start="

[00762] 11 <lookup type="pex" action="get_string" name="title" ref="" location="after" start="<a href="" end="" include_sz="1" strip_tags="1" />

[00763] 12 <lookup type="pex" action="move_ptr" ref="" />

[00764] 13 </element>

[00765] 14 </menu_elements>

[00766] Once the engine has identified that it is looking for a menu on a page that contains the menu style of the "mymenu_1" family the engine finds the spots in the signature file that contains the signature for the objects and elements of the family, shown above.

[00767] <settings>...</settings>

[00768] Contains any page specific manual overrides such as exclude list, customization, modification, personalization, etc. In this example, any result that matches "Site Index", "ReClaim & Insurance Replacement" are excluded but partial matches are also possible by using wild card strings.

[00769] <action>...</action>

[00770] Line 6 and 7 sets the start limit and end limit to instruct the engine on where to look for menu items

[00771] <element>...</element>

[00772] Contains lookup instructions for each object element as previously described. In this example, an element in 'mymenu_1' (each individual menu entry of webpage) contains link and title as its properties. Line 12 instructs the engine to move the pointer to "" to get ready to loop through and extract the next menu item with the same elements

[00773] ABC ComTech Corp. Content/Item Family Signature Explanation

[00774] 1 <item_elements id="myitem_1">

[00775] 2 <actions>

[00776] 3 <lookup type="pex" action="move_ptr" ref="</head>" />

[00777] 4 </actions>

[00778] 5 <element>

[00779] 6 <lookup type="pex" action="get_string" name="image" ref="largeimageref" location="after" start="

[00780] 7 <lookup type="pex" action="get_string" name="title" ref="product-details-prd-title" location="after" start="<span" end="" include_sz="1" strip_tags="1" />

[00781] 8 <lookup type="pex" action="get_string" name="price" ref="our price:" location="after" start="<td" end="</td>" include_sz="1" strip_tags="1" />

[00782] 9 <lookup type="pex" action="get_string" name="sale_price" ref="sale price:" location="after" start="<td" end="</td>" include_sz="1" strip_tags="1" tolerance="1" />

[00783] 10 <lookup type="pex" action="get_string" name="description" ref="detailbox-text" location="middle" start="<p" end="</p>" include_sz="1" strip_tags="1" />

[00784] 11 </element>

[00785] 12 </item_elements>

[00786] Once the engine has identified that the page is of the “myitem_1” family the engine finds the spots in the signature file that contains the signature for the objects and elements of the family, shown above.

[00787] <action>...</action>

[00788] Instructs the engine to move the scan pointer to the appropriate spot to get ready to scan and output the product elements.

[00789] <element>...</element>

[00790] Contains lookup instructions for each of the defined fields in a product. In this example, an object in ‘myitem_1’ (an item) contains the elements image, title, price, sale price and description. Note that the pointer does not need to be moved after scanning in the

elements since there are no more objects on the product detail page as there are on the list page.

[00791] This family had a detail walk though explained above

[00792] Appendix: Signature Engine Syntax

[00793] Lookup Syntax

[00794] Look up is the query which Signature engine runs against the website for a resultset(s).

[00795] Type

[00796] Defines data type of reference. One lookup can contain multiple references and specific type of each reference, represented by 'Type_n' for each nth reference.

Type	Description
Pex	String expression
Iex	Numeric expression
Rex	Regular expression
Dex	Date/Time expression
Bex	Binary

[00797] Action

Action	SQL Equivalent
Locate_string	SELECT COUNT ALL
Get_string	SELECT
Move_ptr	Beginning_of_LIMIT
End_ptr	End_of_LIMIT
Remove_string	DELETE
Replace_string	UPDATE

[00798] Name

[00799] An element of an object, for example price if the object is a product

[00800] Id

[00801] Id is a named relation for an identified family of web pages. It is the SQL equivalent of a table name.

[00802] Ref

[00803] Ref is the string reference being matched to identify a page, object or element. It is equivalent to a WHERE condition in SQL. There could be multiple Ref values where default Ref is represented by 'Ref' and subsequent Refs are presented by 'Ref_n'. This is equivalent to having multiple conditions in an SQL WHERE clause.

[00804] Alt

[00805] Alt is an extension of Ref object with SQL equivalent of 'OR' clause presented by 'Ref_Alt_n'.

[00806] Location

[00807] Extends SQL equivalent of WHERE clause to give directional containment of data set.

Location	Description
Before	Target value is located before 'Ref' value
Middle	Target value contains 'Ref' value
After	Target value is located after 'Ref' value

[00808] Start

[00809] Start is used to specify the beginning of a given data element.

[00810] End

[00811] End is used to specify end of a given data element.

[00812] Include_sz

[00813] Boolean to include 'Start' and 'End' value.

[00814] Tolerance

[00815] Boolean to define whether failure of given lookup excludes the entire query for an object.

[00816] Strip_tags

- [00817]** Boolean to define where to strip all HTML tags out of the target value being extracted.
- [00818]** Notrim
- [00819]** Values will not be trimmed (leading and trailing spaces will not be removed).
- [00820]** Upper
- [00821]** Value is converted to all upper case.
- [00822]** Lower
- [00823]** Value is converted to all lower case.
- [00824]** Uppercase_word
- [00825]** First character of each words in a value is converted to upper case.
- [00826]** Uppercase_first
- [00827]** First character of a value is converted to upper case.
- [00828]** Page Syntax
- [00829]** Make a page family's paging feature functional.
- [00830]** Page_variable
- [00831]** Defines unique key that defines a family's paging feature
- [00832]** Page_start
- [00833]** Defines value of first page in a family's paging feature.
- [00834]** Page_post
- [00835]** Path where paging variable(s) must be transmitted to.
- [00836]** Page_start
- [00837]** Defines value of first page in a family's paging feature.

[00838] Search Syntax

[00839] Make a website family's search feature functional.

[00840] Search_path

[00841] Search path where search variable must be transmitted to

[00842] Search_variable

[00843] Name of search variable which a website's search feature is looking to read, request, post, etc.

[00844] General Syntax

[00845] Any variable name and value can be defined.

[00846] TOR_LAW\ 6600992\1

CLAIMS

1. A system for content navigation comprising:

a network;

a first computing device configured to communicate over said network; said first computing device having a native menu application and a browser application;

at least one additional computing device configured to communicate over said network and to provide interactive content; said browser application configured to access said interactive content; said interactive content comprising a plurality of pages; the plurality of said pages including a menu section and a content section; each said menu section comprising addresses to one or more of said pages;

said at least one additional computing device further configured to provide a schema corresponding to said interactive content; said first computing device configured to access said schema; said schema comprising menu instructions for said native menu application; said menu instructions corresponding at least in part to addresses associated with each said menu section;

said first computing device configured to utilize said schema such that said browser application is configured to generate said content sections and said native menu application is configured to generate said menu instructions on said first computing device such that said pages can be navigated using said native menu application.

2. The system of claim 1 wherein said network is the Internet and said first computing device is a mobile electronic device having combined functionality of a personal digital assistant, cell phone, email paging device, and a web-browser configured to communicate over the Internet.

3. The system of claim 1 or 2 wherein said native menu application is further configured to generate control options in addition to said menu instructions.

4. The system of claim 3 wherein said control options include at least one of an instruction to: move back one page in said browser application; move forward one page in said browser application; close said browser application; and close said

menu application.

5. The system of any of claims 1-4 wherein said network comprises the Internet and said interactive content is a web-site.
6. The system of any of claims 1-5 wherein said schema is generated in eXtended Markup Language format.
7. The system of any of claims 1-6 wherein said at least one additional computing device comprises a web-server for hosting said interactive content and a schema server for hosting said schema.
8. The system of claim 7 wherein said first computing device is configured to maintain a network address for said schema server.
9. The system of claim 7 wherein said schema server is further configured to transcode each said content section for optimized generation on a display of said first computing device.
10. The system of any of claims 1-9 wherein said browser application includes a transcoding engine.
11. A method of content navigation in a computing device comprising:
 - maintaining a browser application;
 - maintaining a native menu application;
 - receiving a portion of interactive content; said interactive content comprising a plurality of pages; the plurality of said pages including a menu section and a content section; each said menu section comprising addresses to one or more of said pages;
 - receiving a schema corresponding to said interactive content; said schema comprising menu instructions for said native menu application; said menu instructions corresponding at least in part to addresses associated with each said menu section;
 - generating said portion of said interactive content using said browser application;

generating menu instructions corresponding to said portion of said interactive content using said native menu application.

12. The method of claim 11 wherein said computing device is a mobile electronic device having combined functionality of a personal digital assistant, cell phone, email paging device, and a web-browser configured to communicate over the Internet.
13. The method of claim 11 or claim 12 further comprising generating control options in addition to said menu instructions using said native menu application.
14. The method of claim 13 wherein said control options include at least one of an instruction to: move back one page in said browser application; move forward one page in said browser application; close said browser application; and close said menu application.
15. The method of any of claims 11-14 wherein said interactive content is a web-site.
16. The method of any of claims 11-15 wherein said schema is generated in eXtensible Markup Language format.
17. The method of any of claims 11-16 further comprising transcoding each said content section for optimized generation.
18. A computer readable medium configured to maintain programming instructions in accordance with any of the methods of claims 11-17.
19. A computing device comprising:
 - a network interface configured to communicate over a network;
 - a non-volatile storage unit configured to maintain a native menu application and a browser application;
 - a processor connected to said network interface and said non-volatile storage unit; said processor configured to execute said native menu application and said browser application;
 - said processor configured to communicate with at least one additional computing

device via said network; said at least one additional computing device configured to provide interactive content; said browser application configured to access said interactive content; said interactive content comprising a plurality of pages; the plurality of said pages each including a menu section and a content section; each said menu section comprising addresses to one or more of said pages;

said at least one additional computing device further configured to provide a schema corresponding to said interactive content; said computing device configured to access said schema; said schema comprising menu instructions for said native menu application; said menu instructions corresponding at least in part to addresses associated with each said menu section;

said processor configured to utilize said schema;

a display connected to said processor; said processor configured to control said display; said browser application configured to generate said content sections and said native menu application configured to generate said menu instructions on said display such that said pages can be navigated using said native menu application.

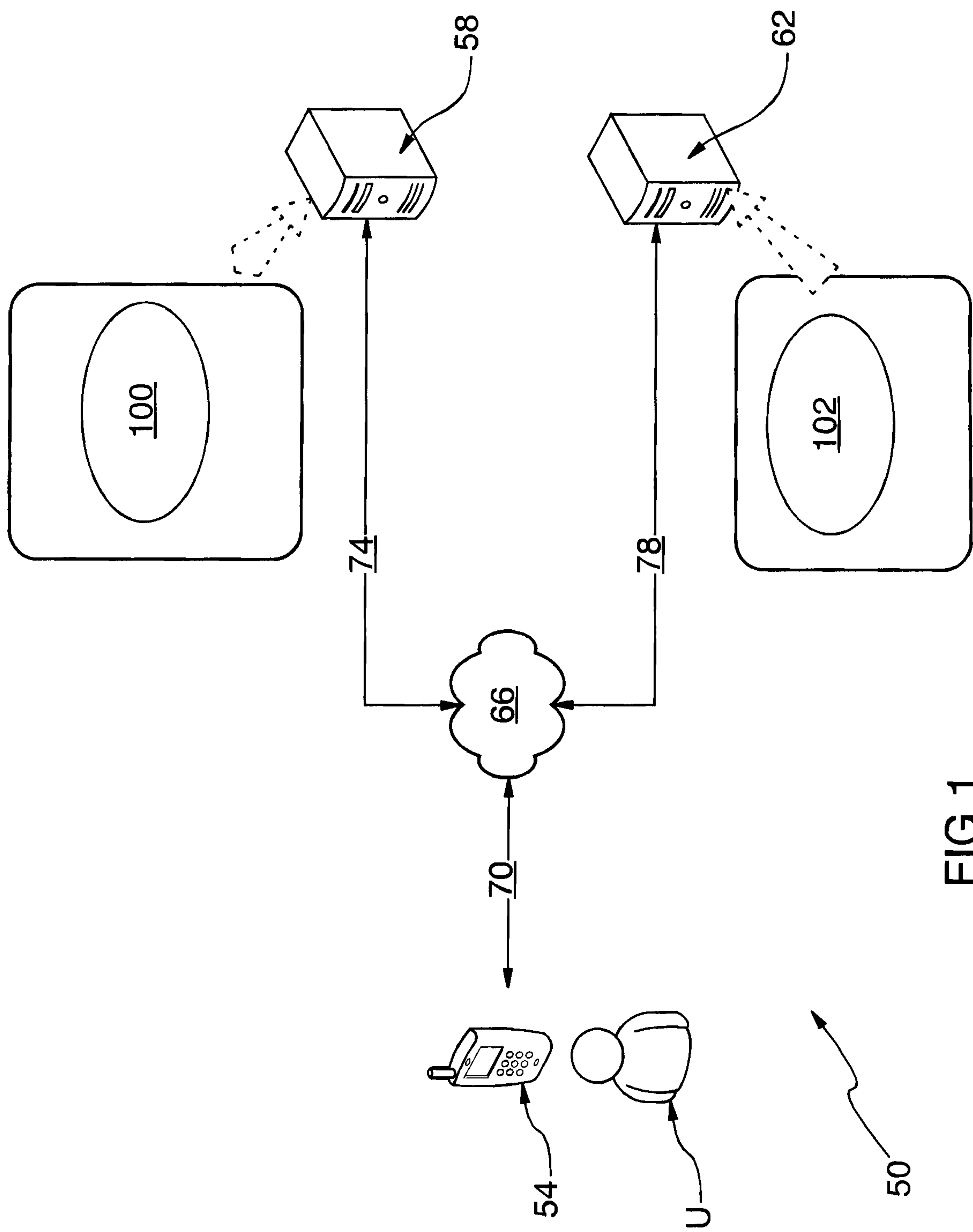


FIG.1

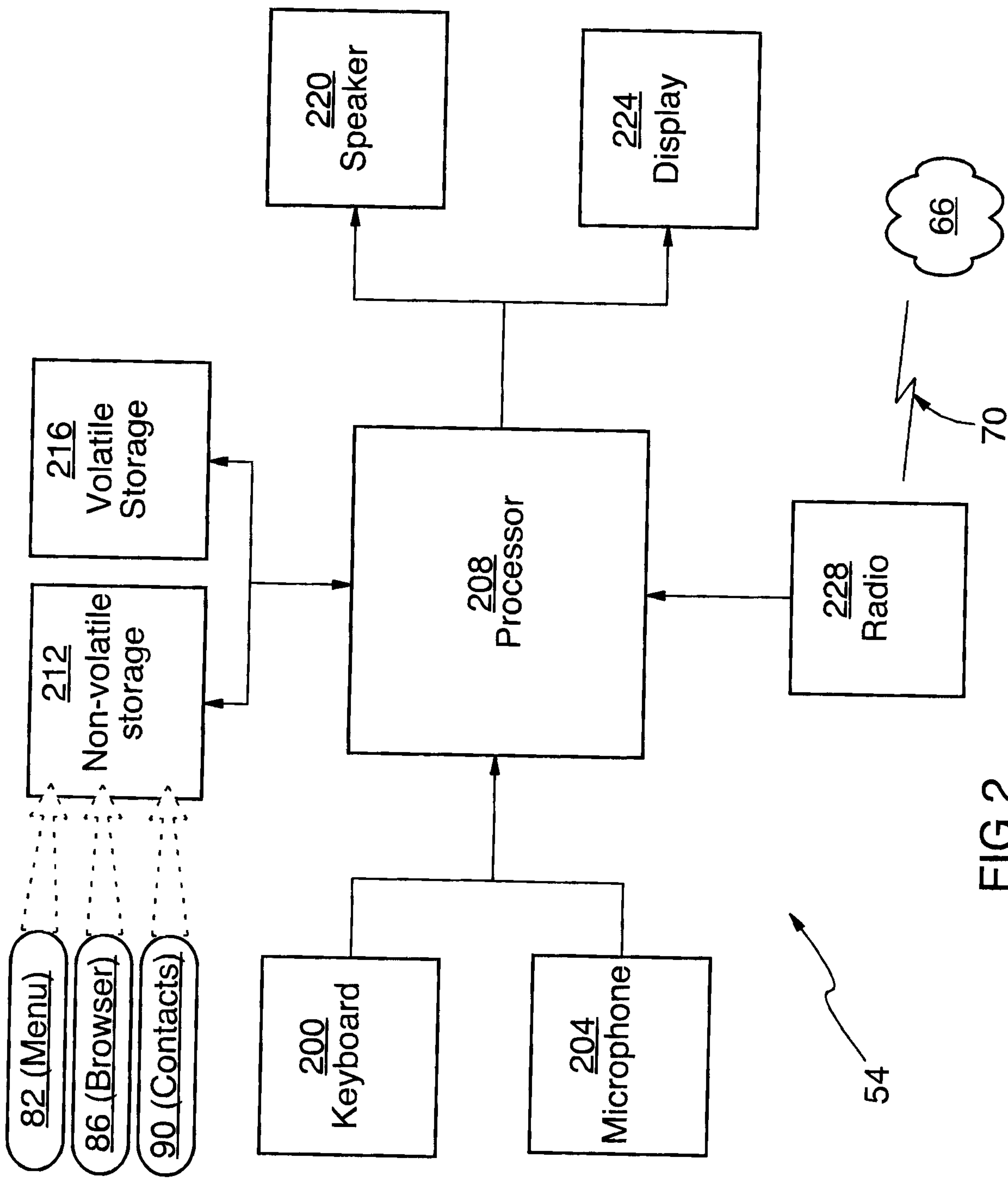


FIG.2

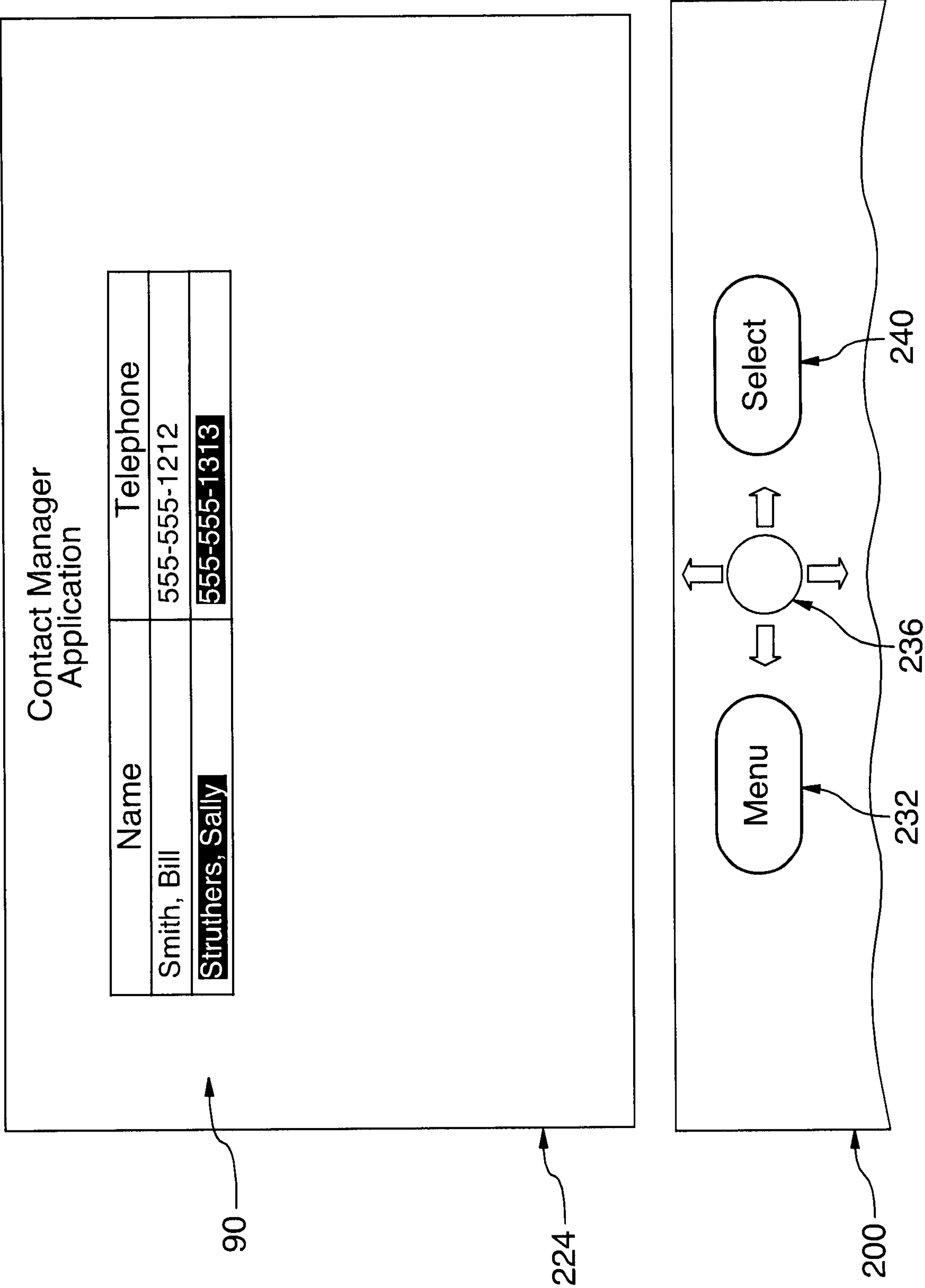


FIG.3

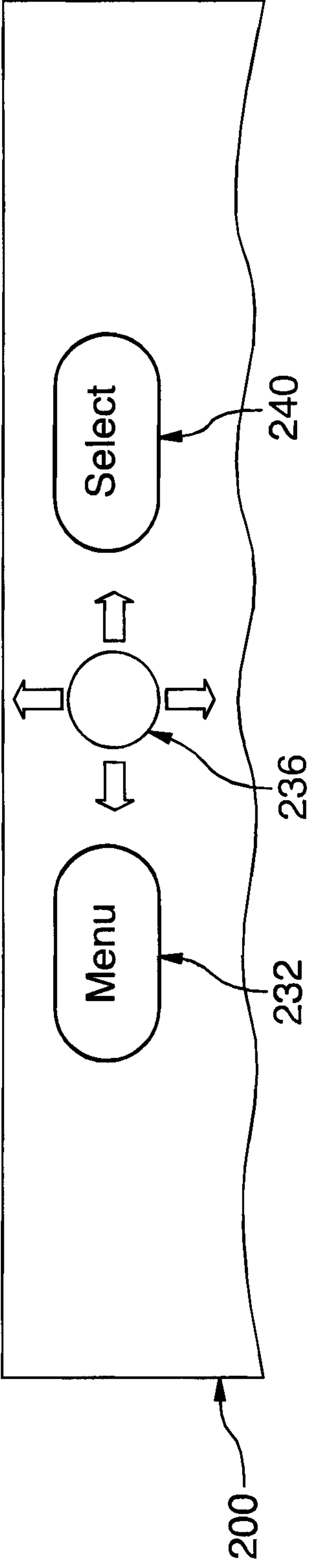
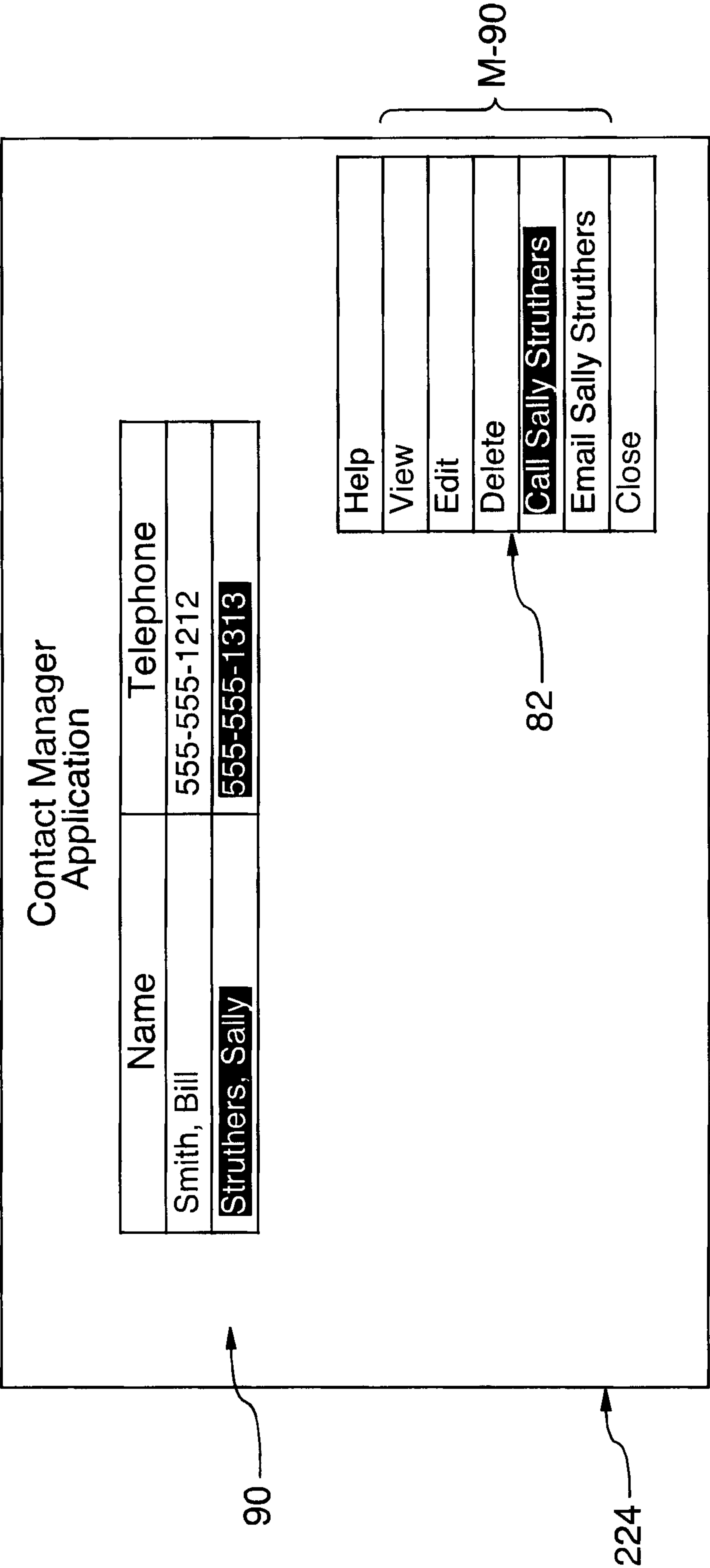


FIG.4

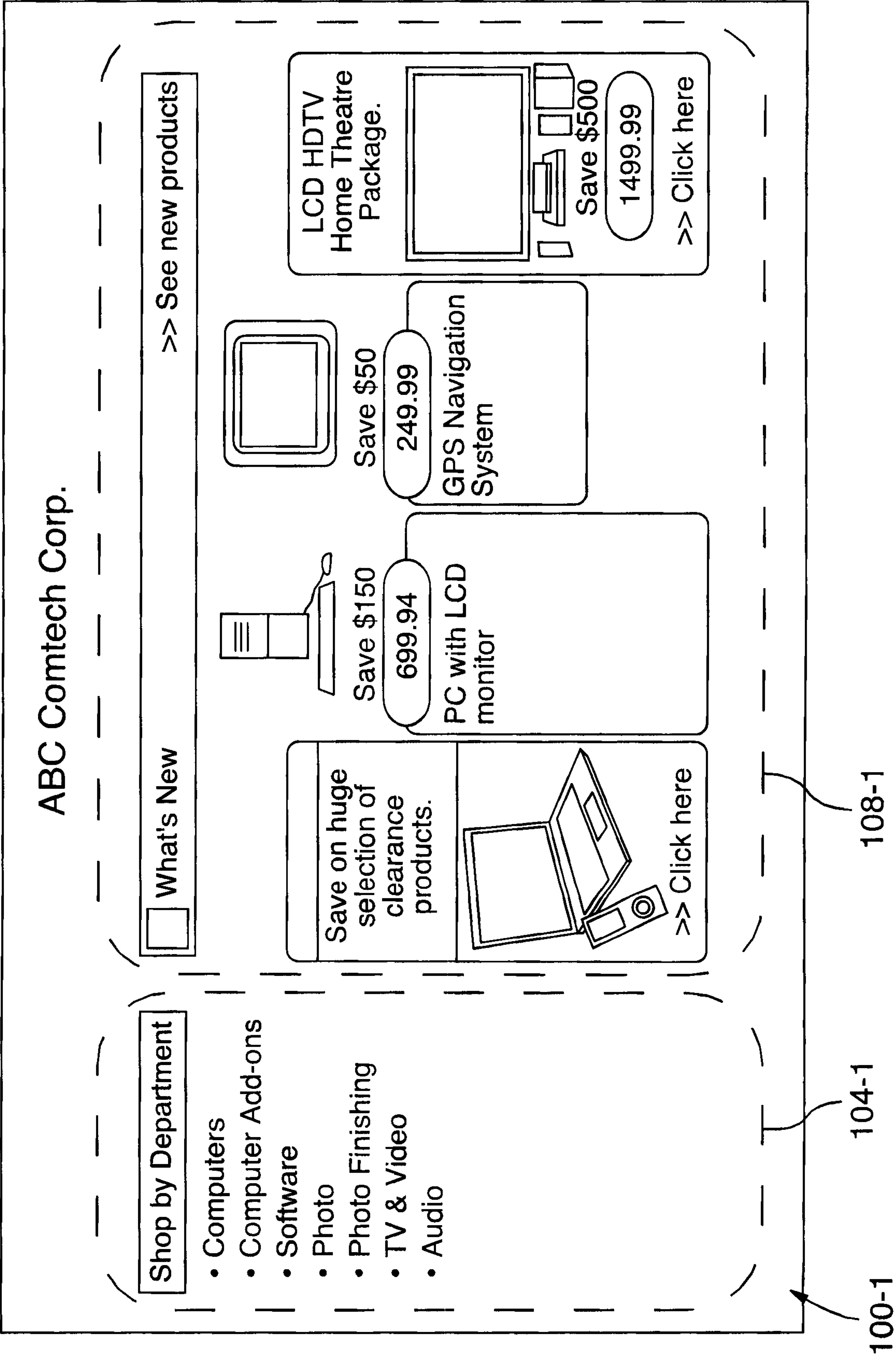


FIG.5

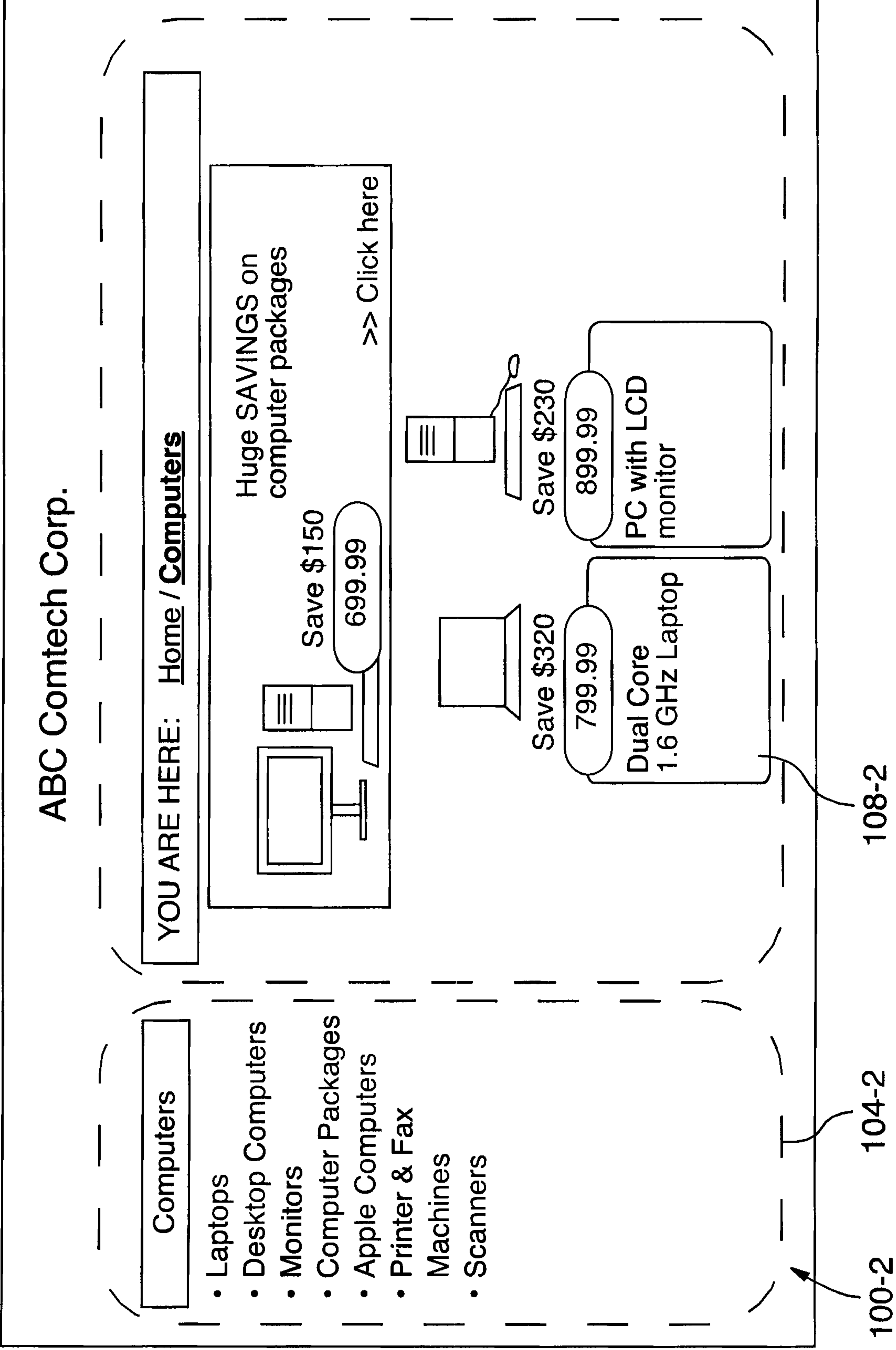


FIG.6

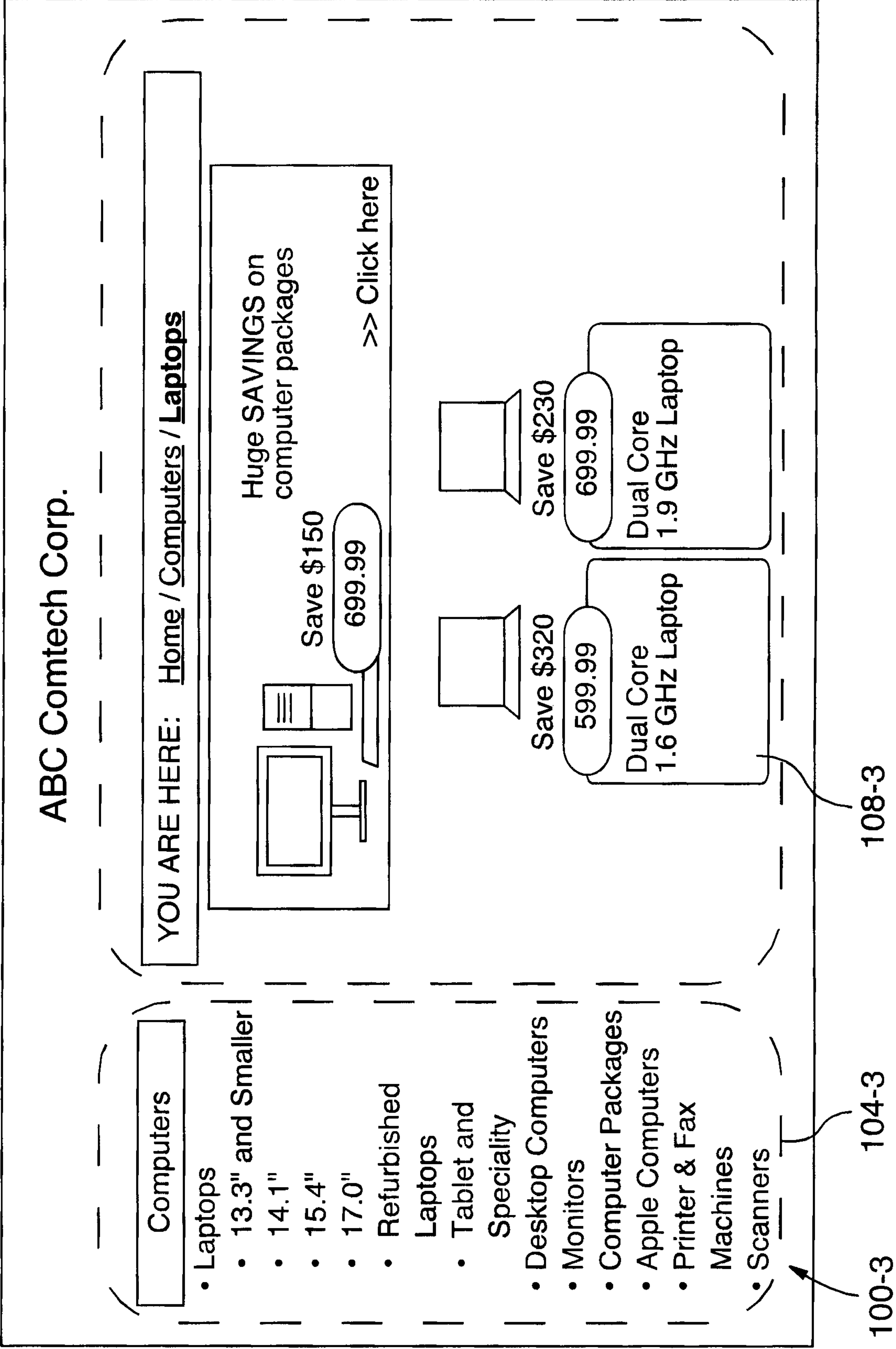


FIG.7

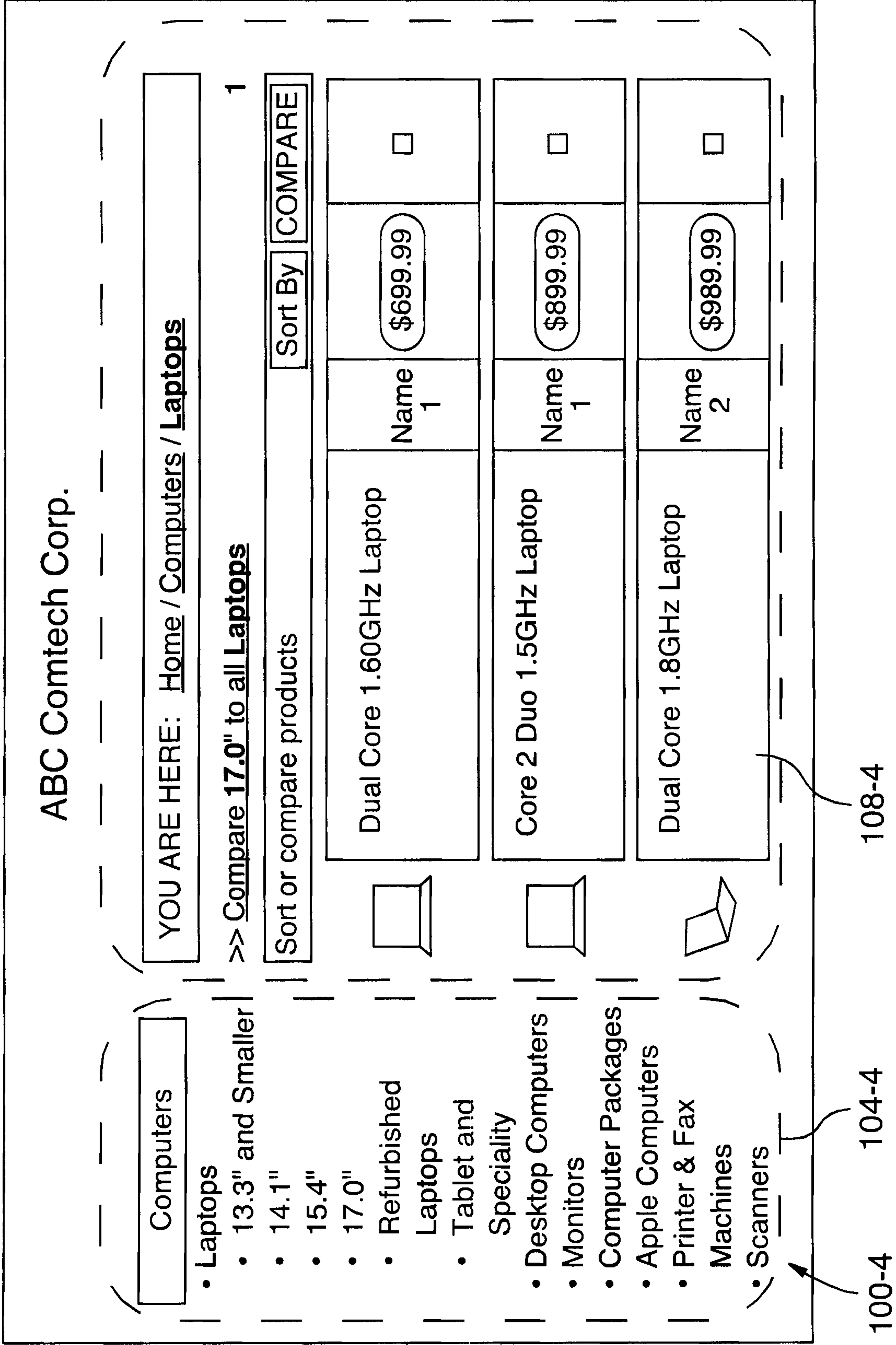


FIG.8

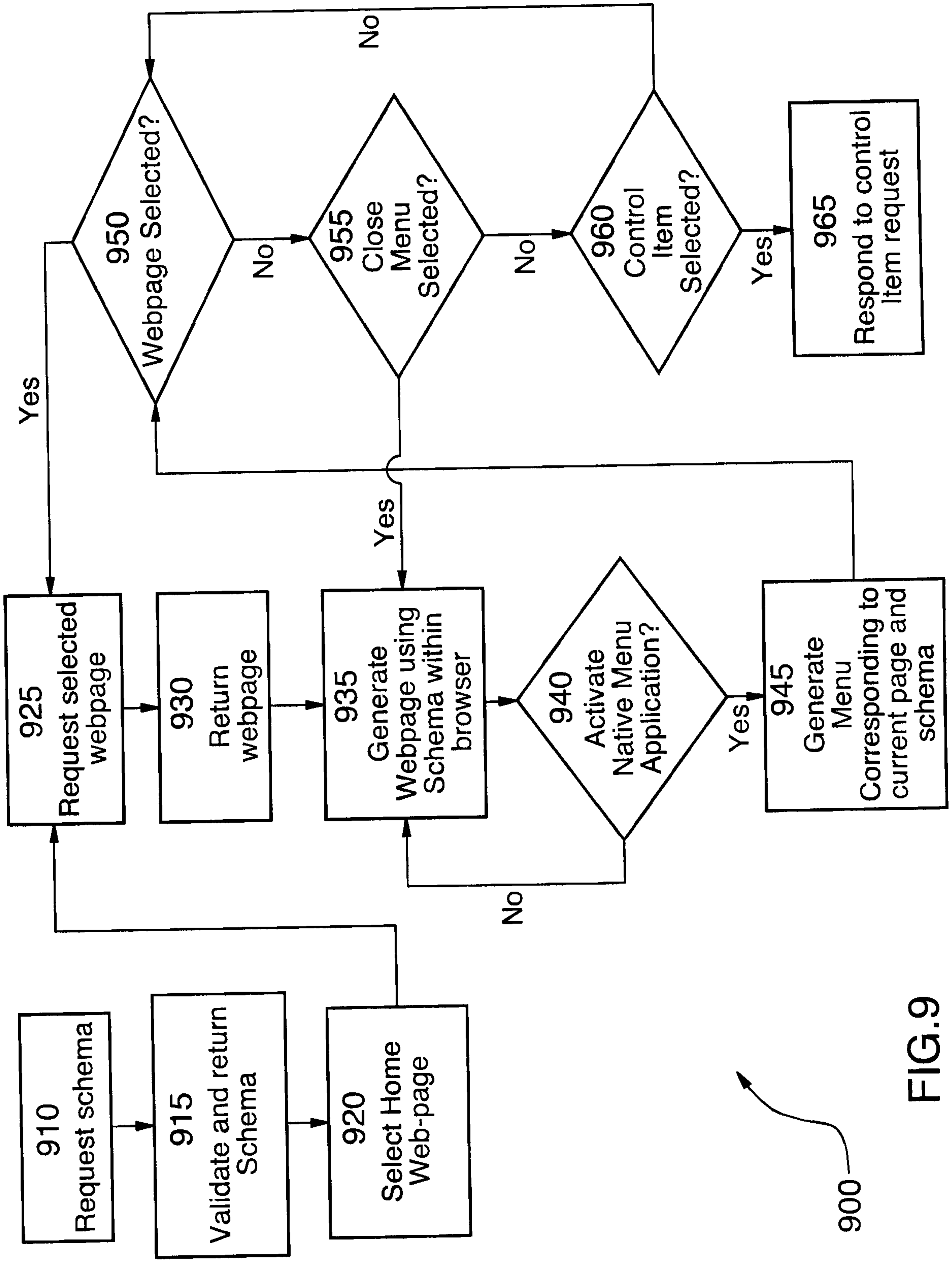
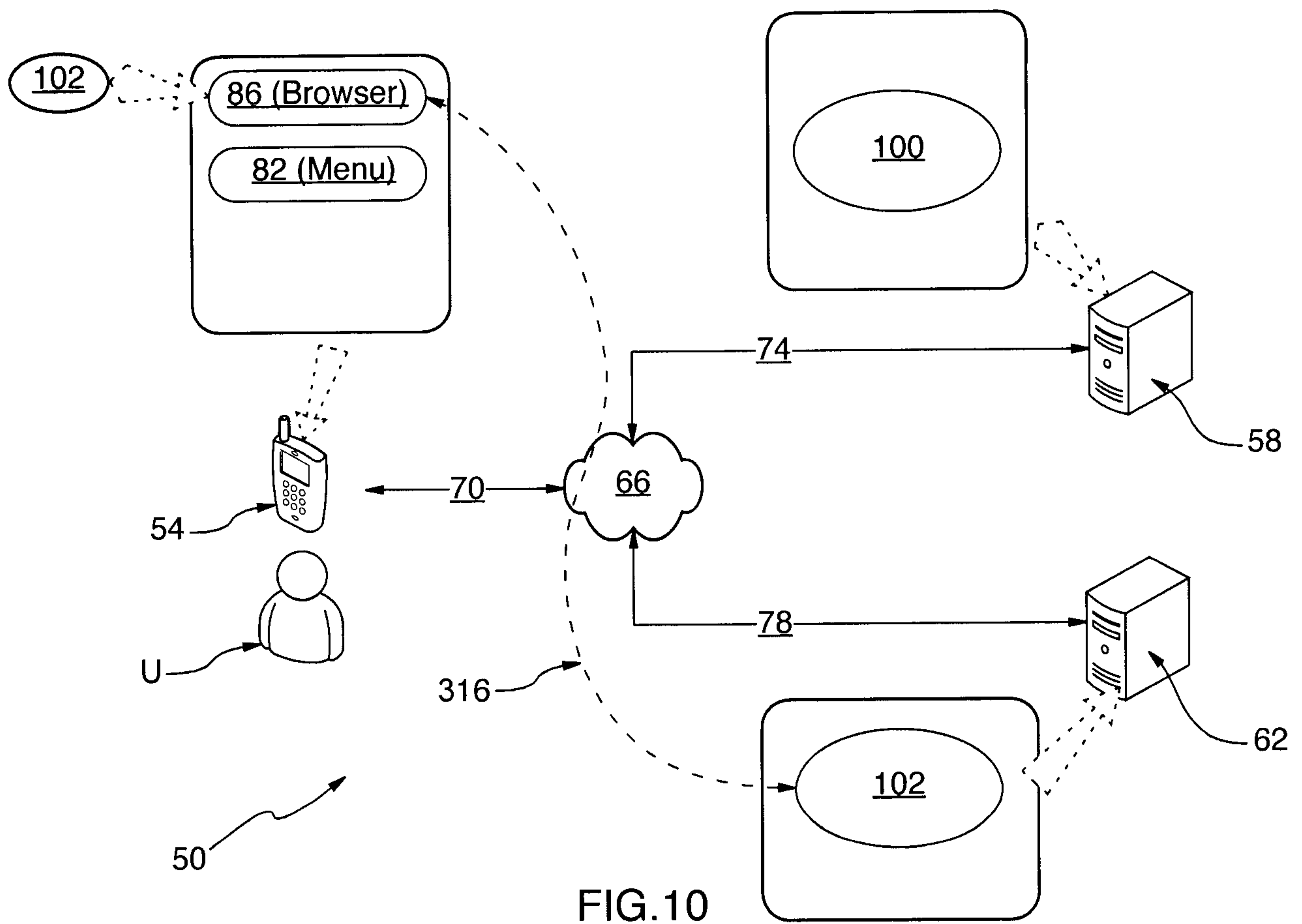


FIG.9



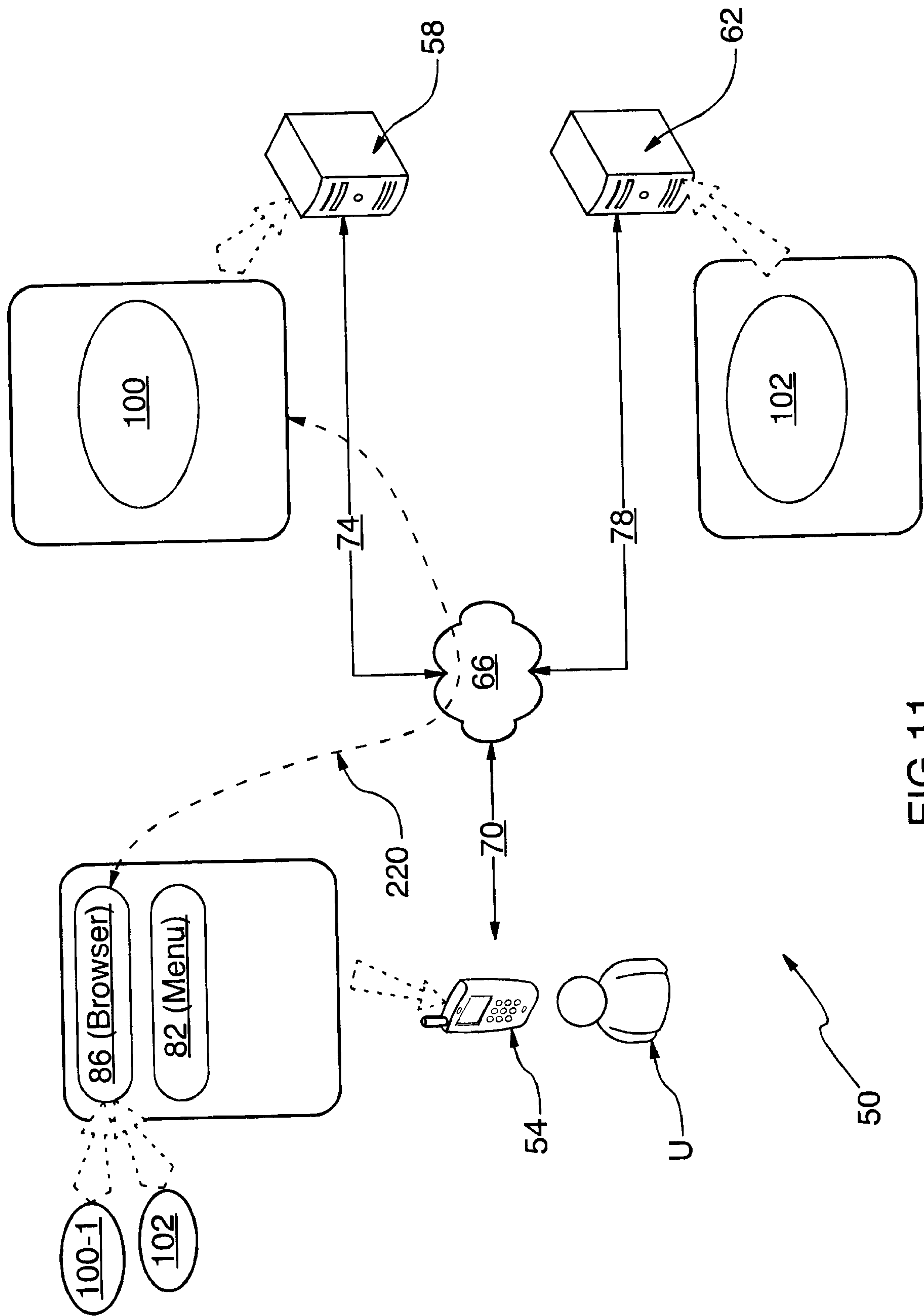


FIG.11

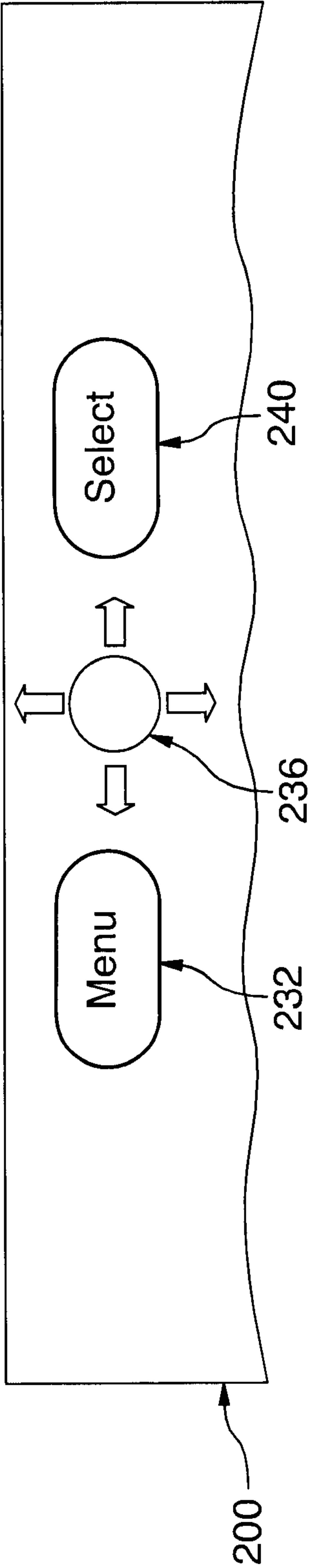
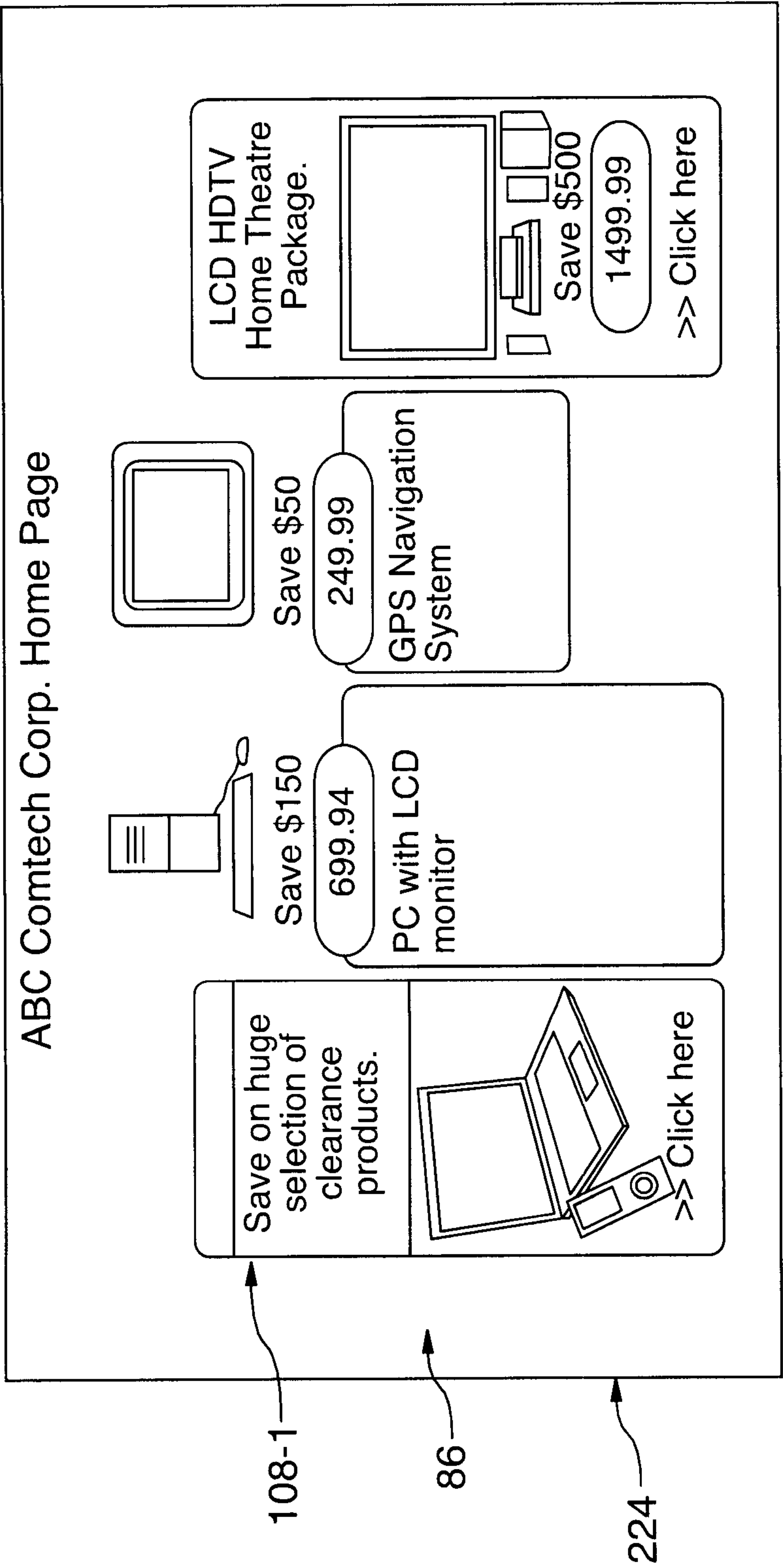


FIG.12

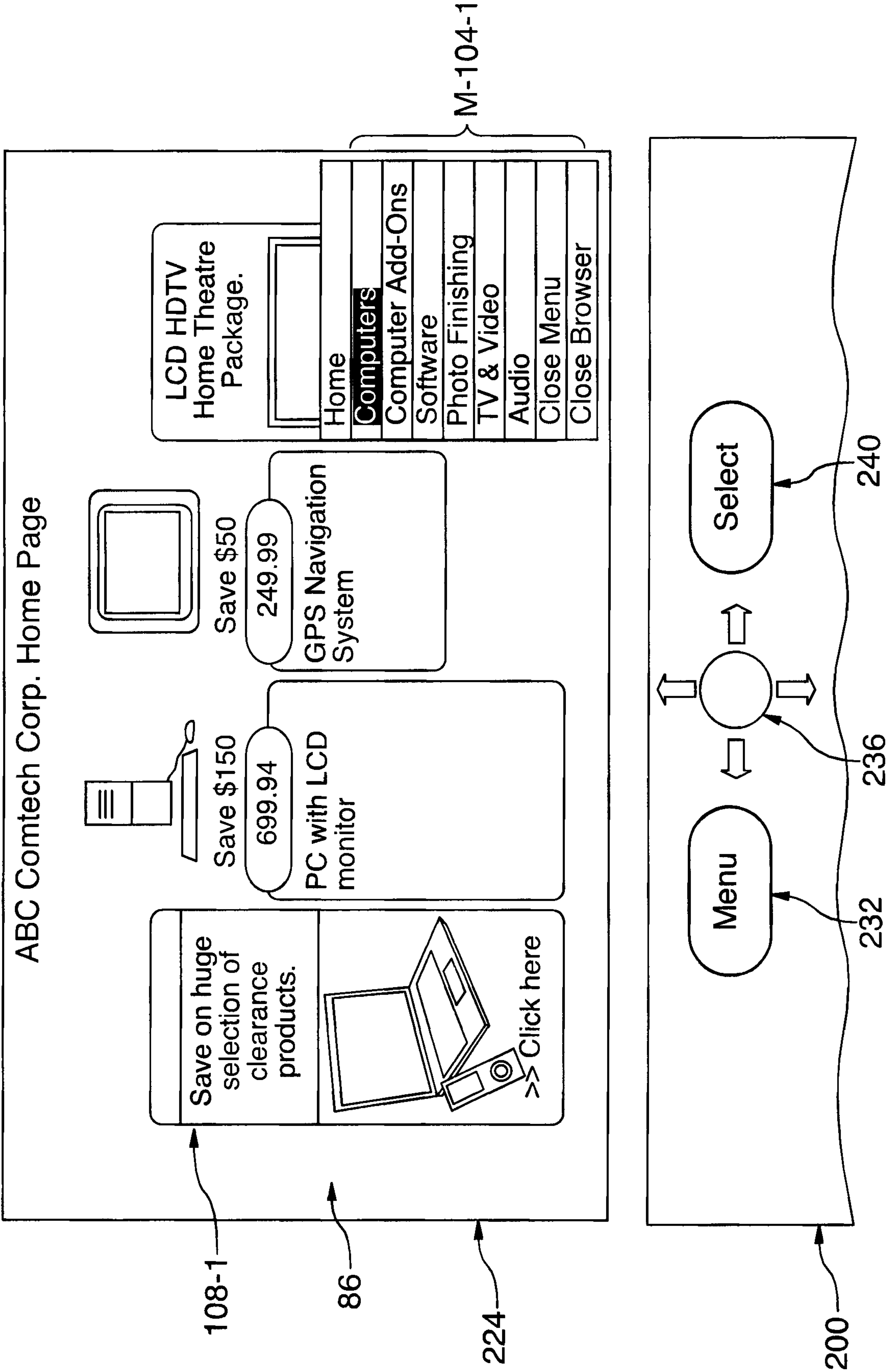


FIG.13

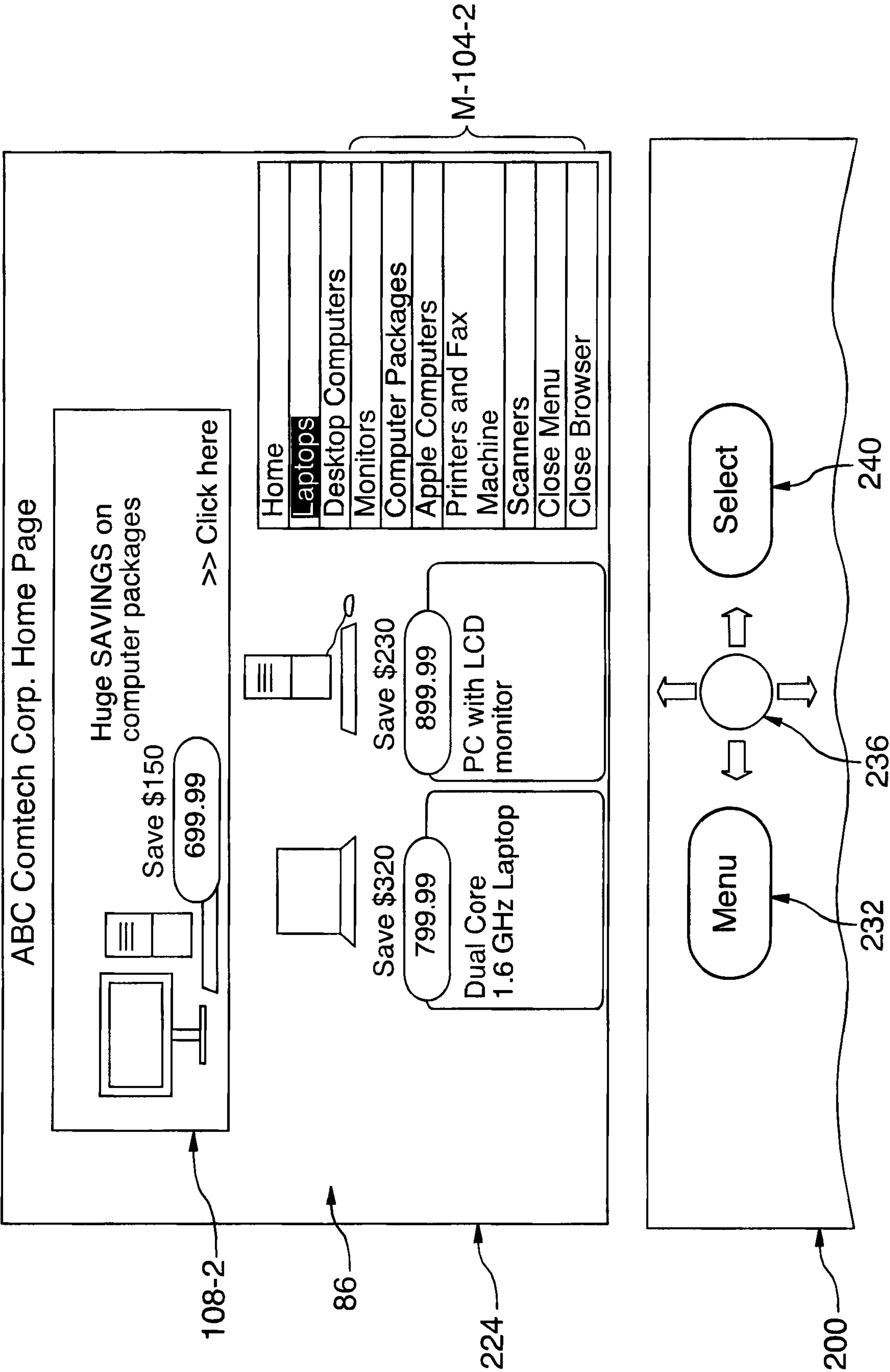


FIG.14

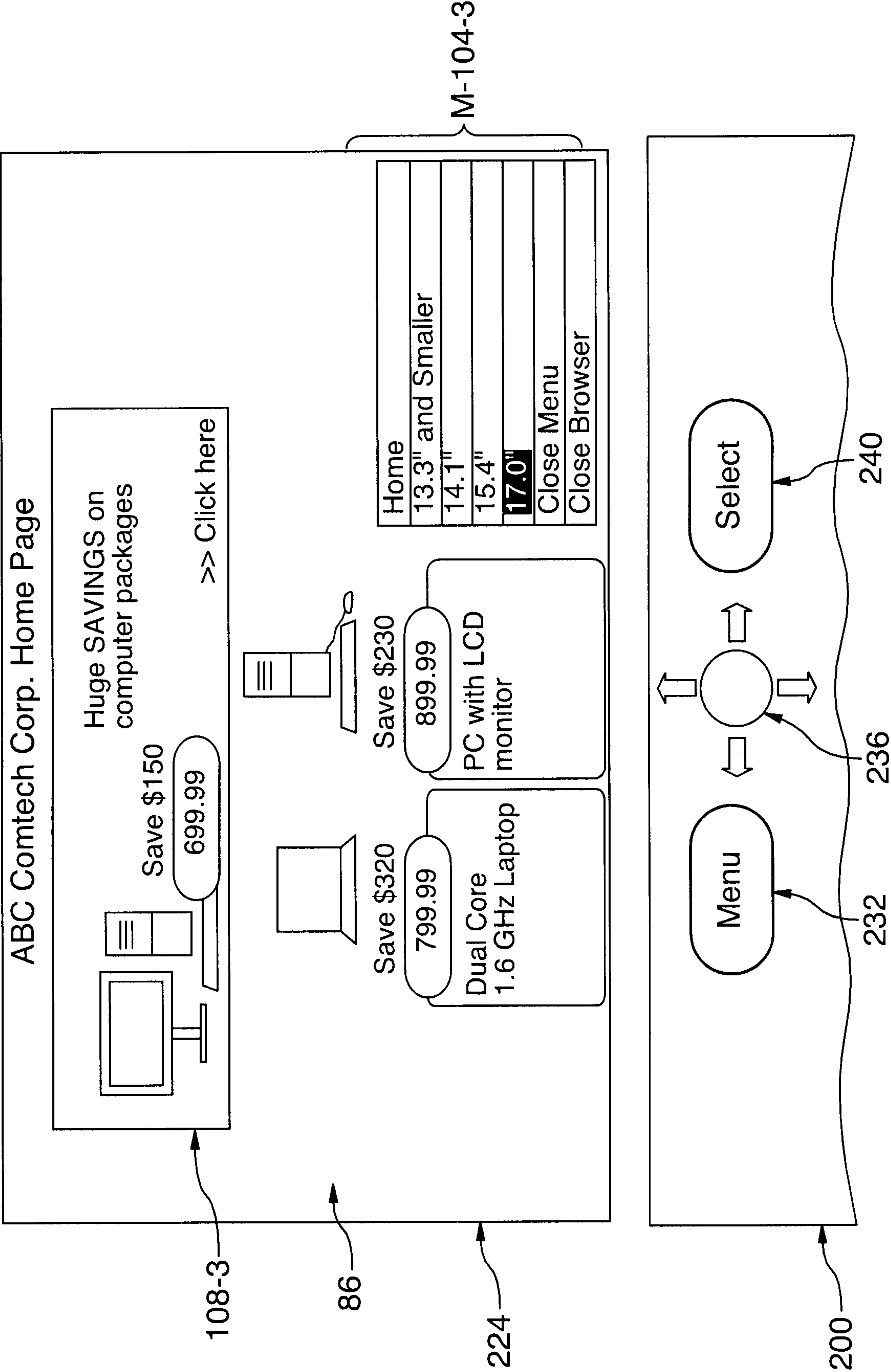
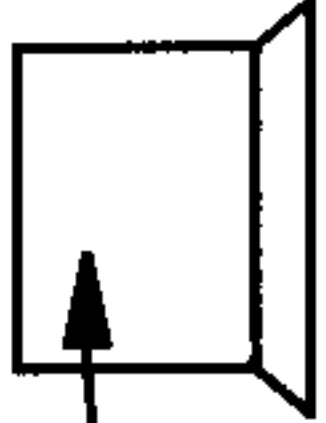


FIG.15

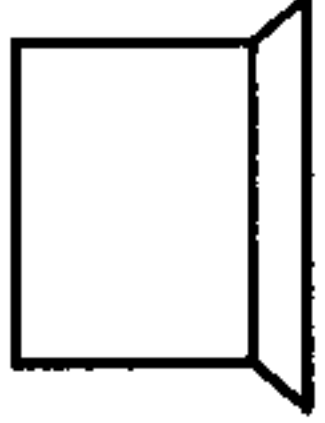
108-4



ABC Comtech Corp. 17.0" Laptop Page

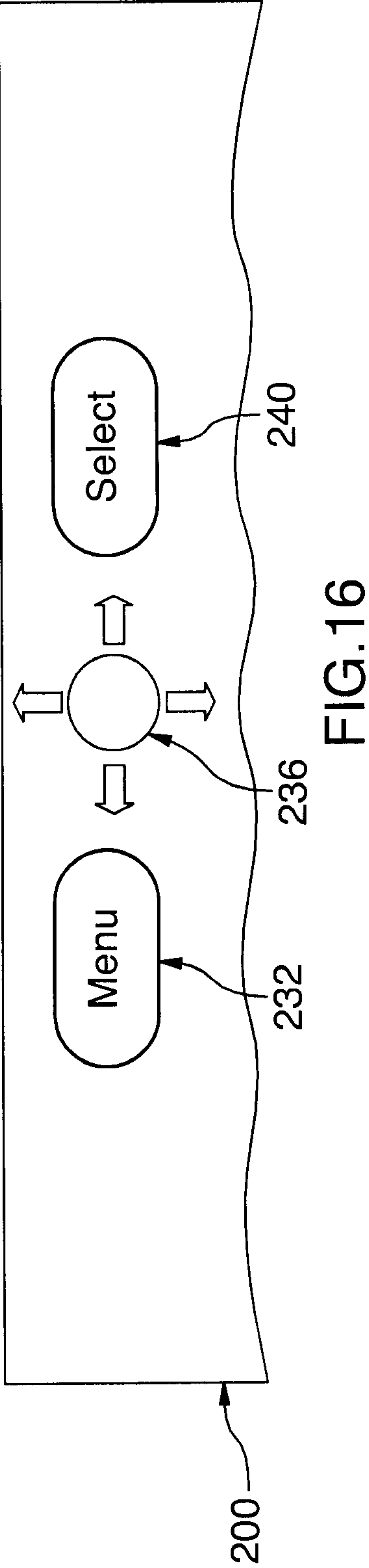
Dual Core 1.60GHz Laptop	Name ₁	\$699.99	☐
Core 2 Duo 1.5GHz Laptop	Name ₁	\$899.99	☐
Dual Core 1.8GHz Laptop	Name ₂	\$989.99	☐

86



224





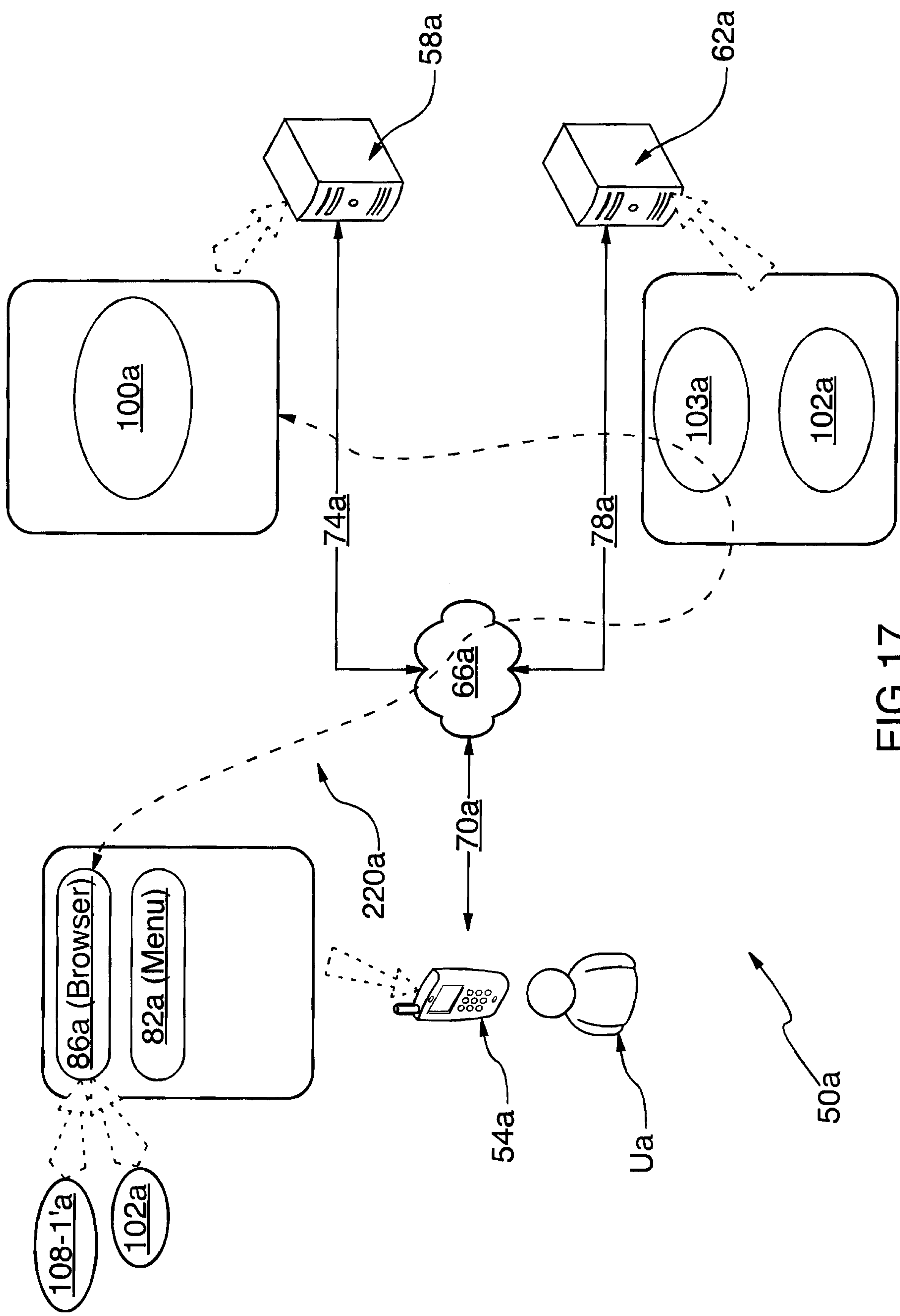


FIG.17

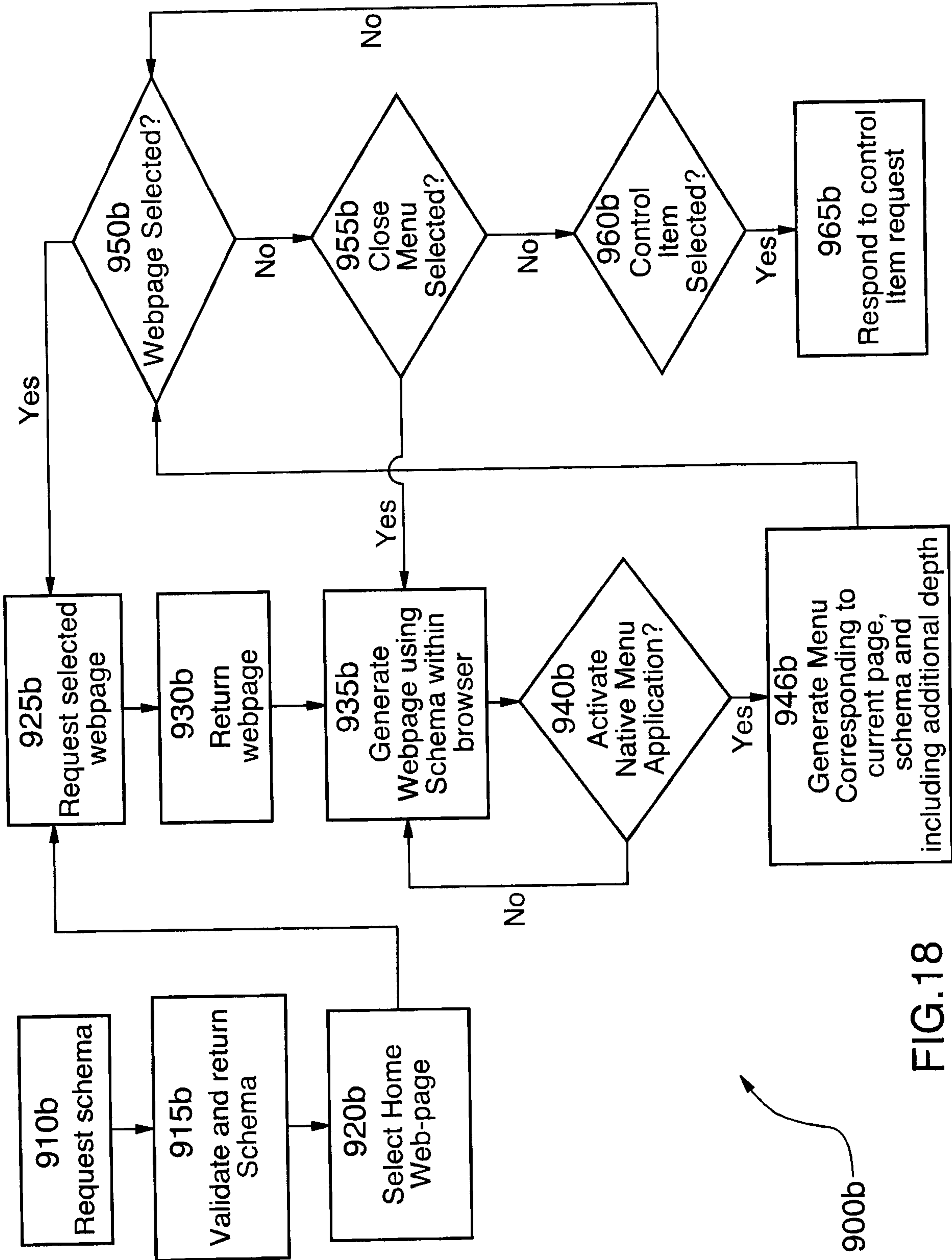


FIG.18

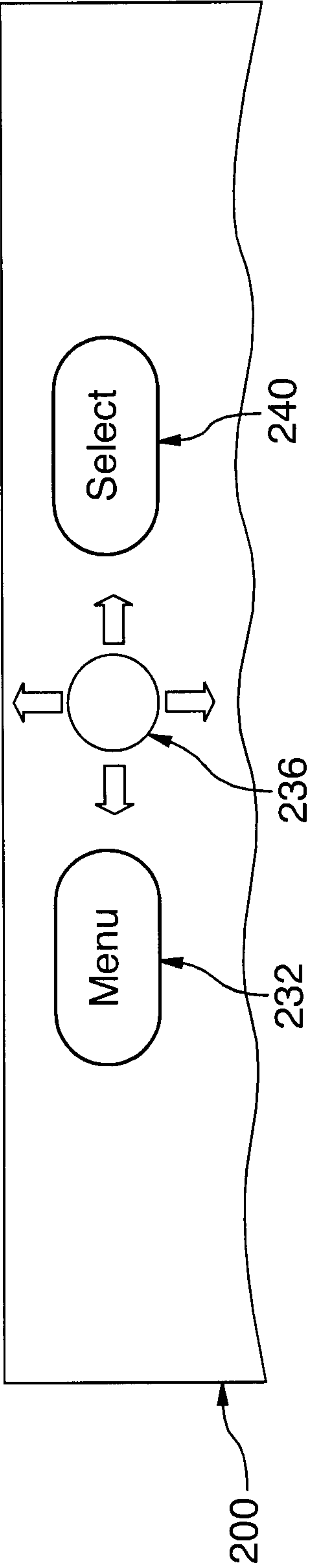
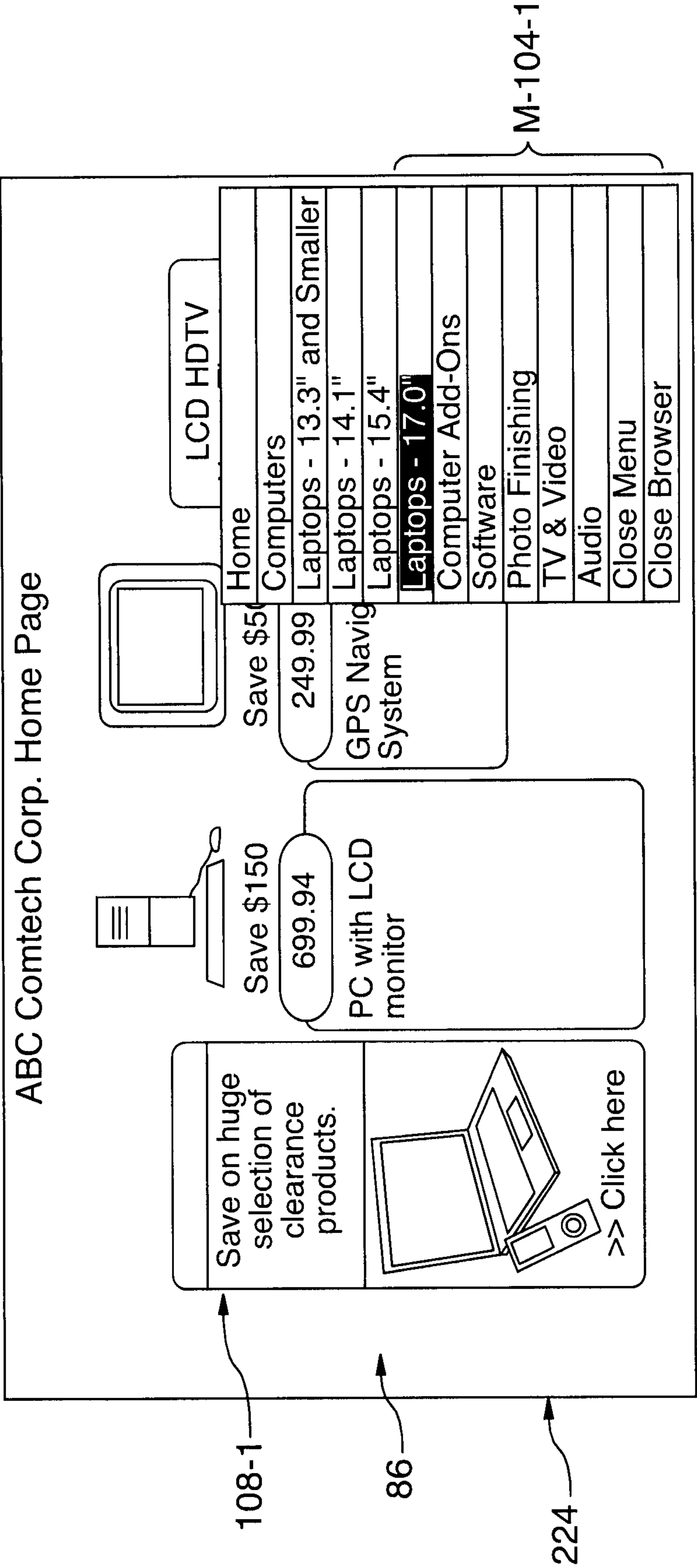


FIG.19

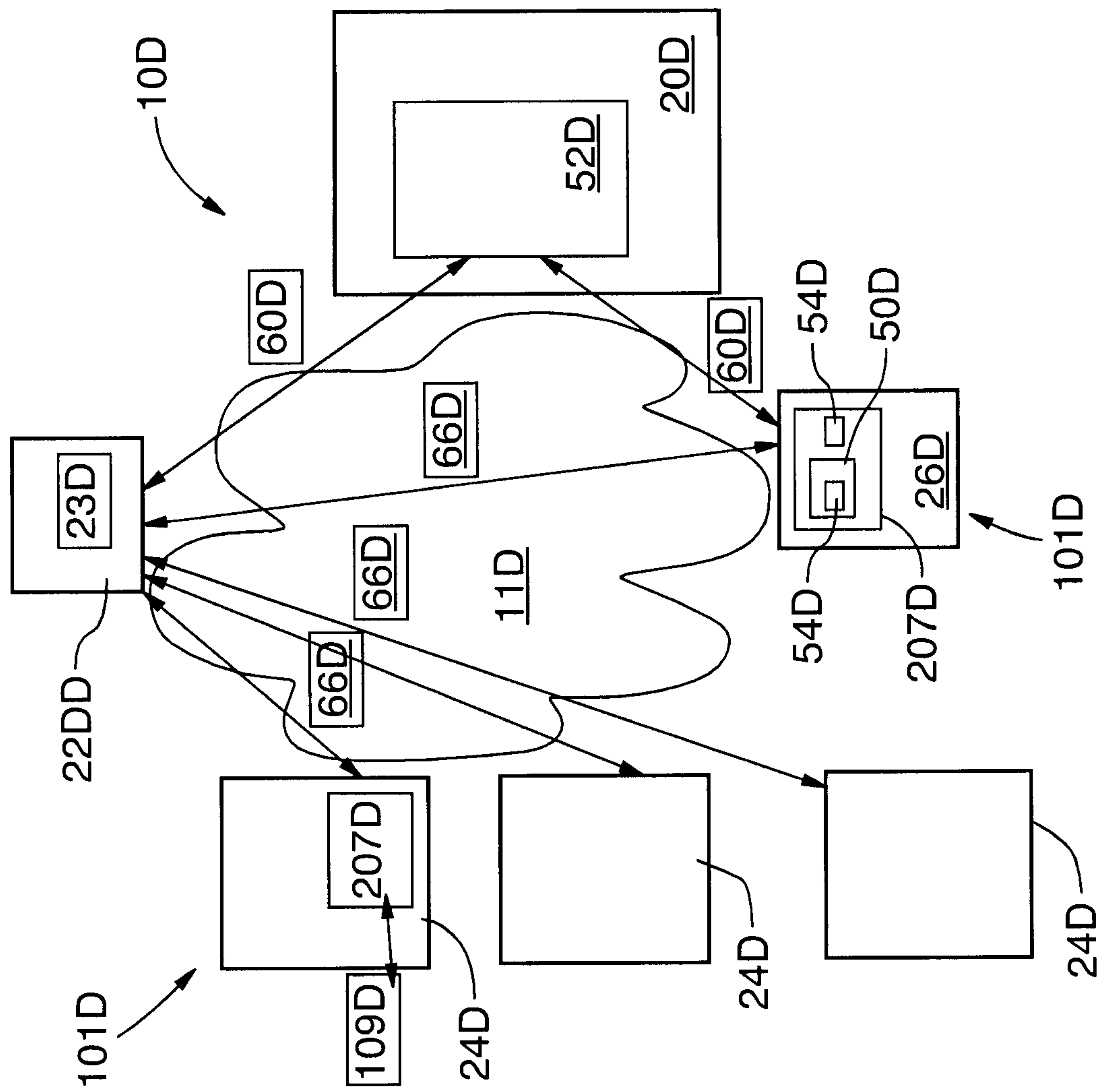


FIG.20

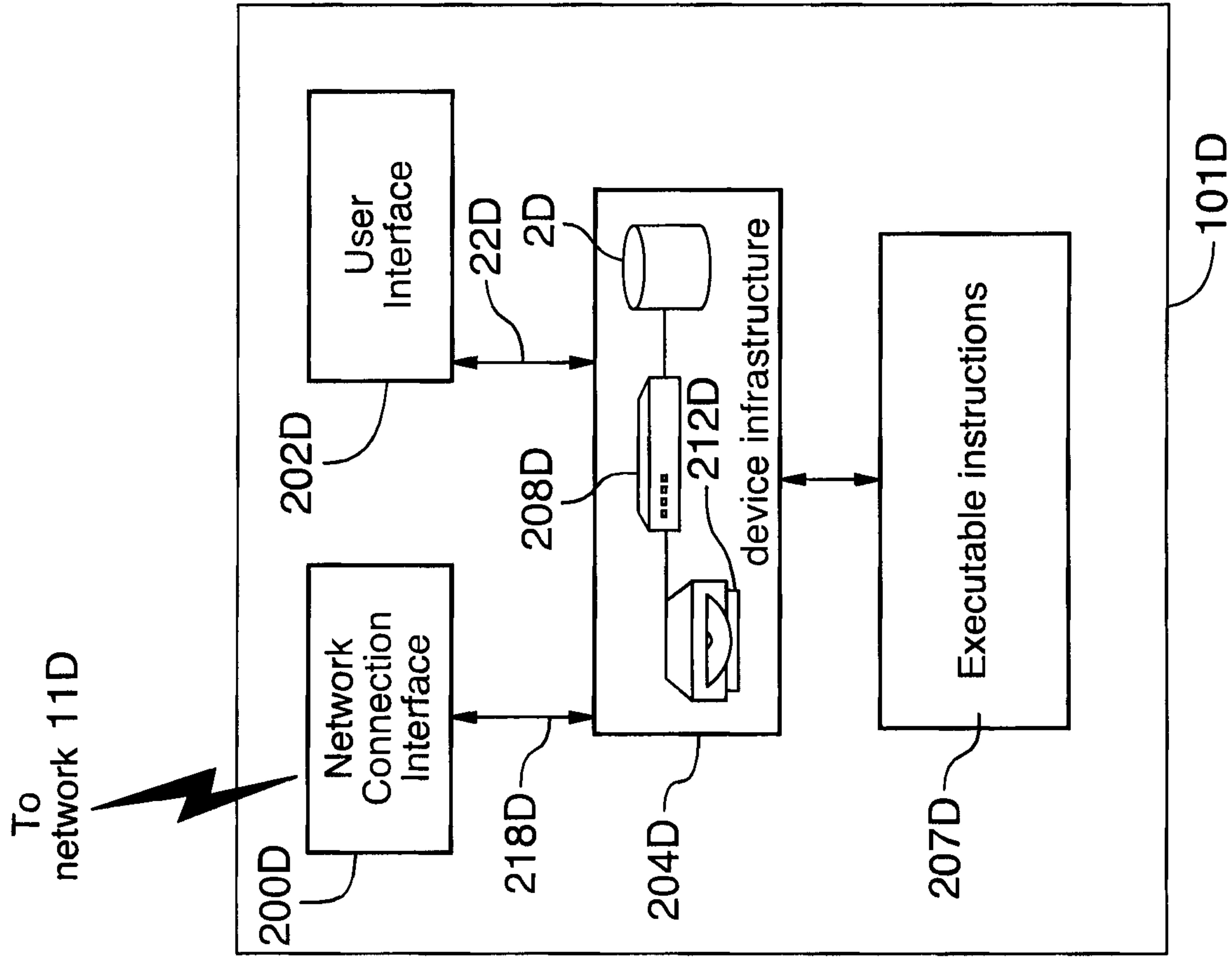


FIG.21

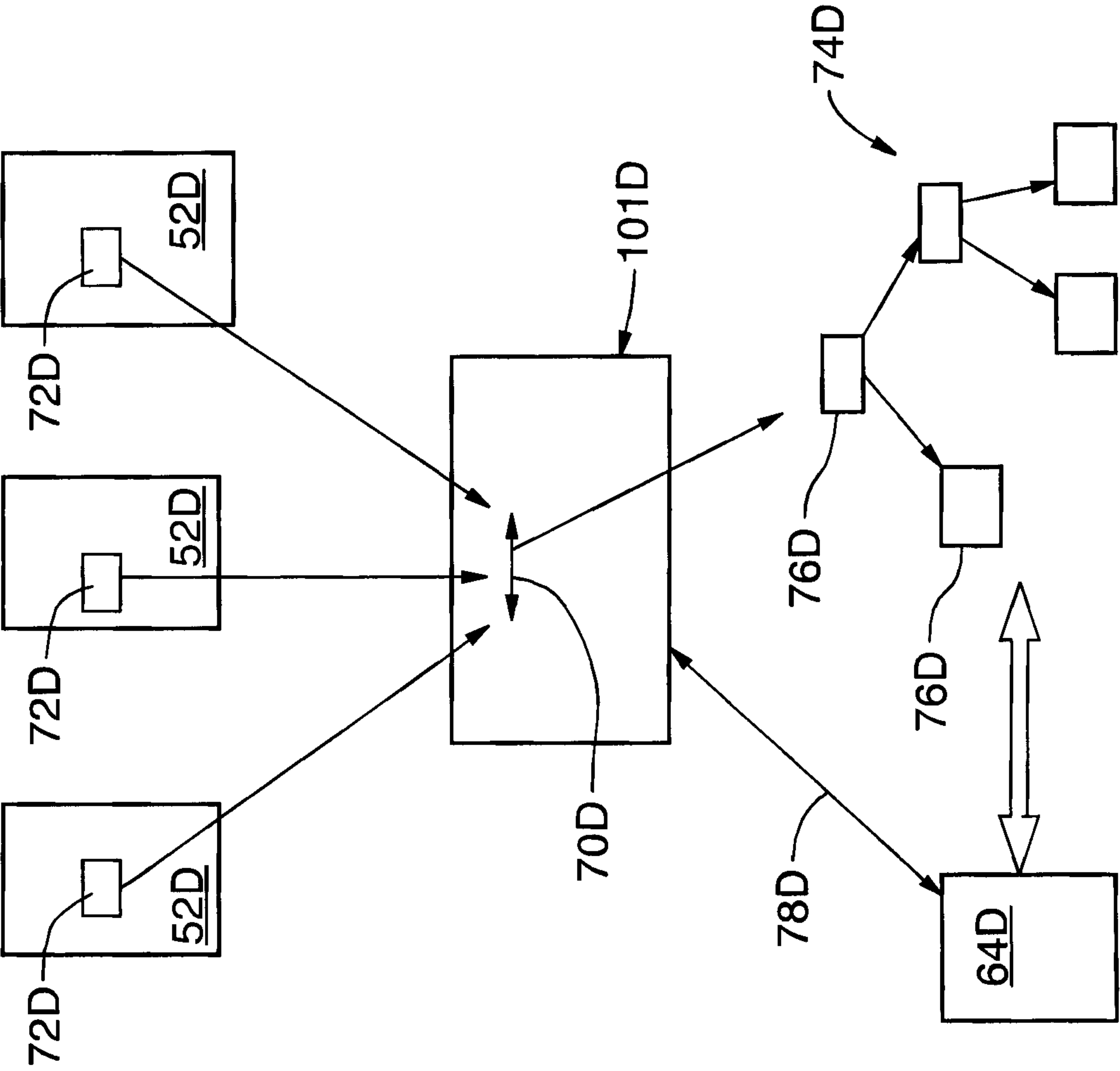


FIG.22

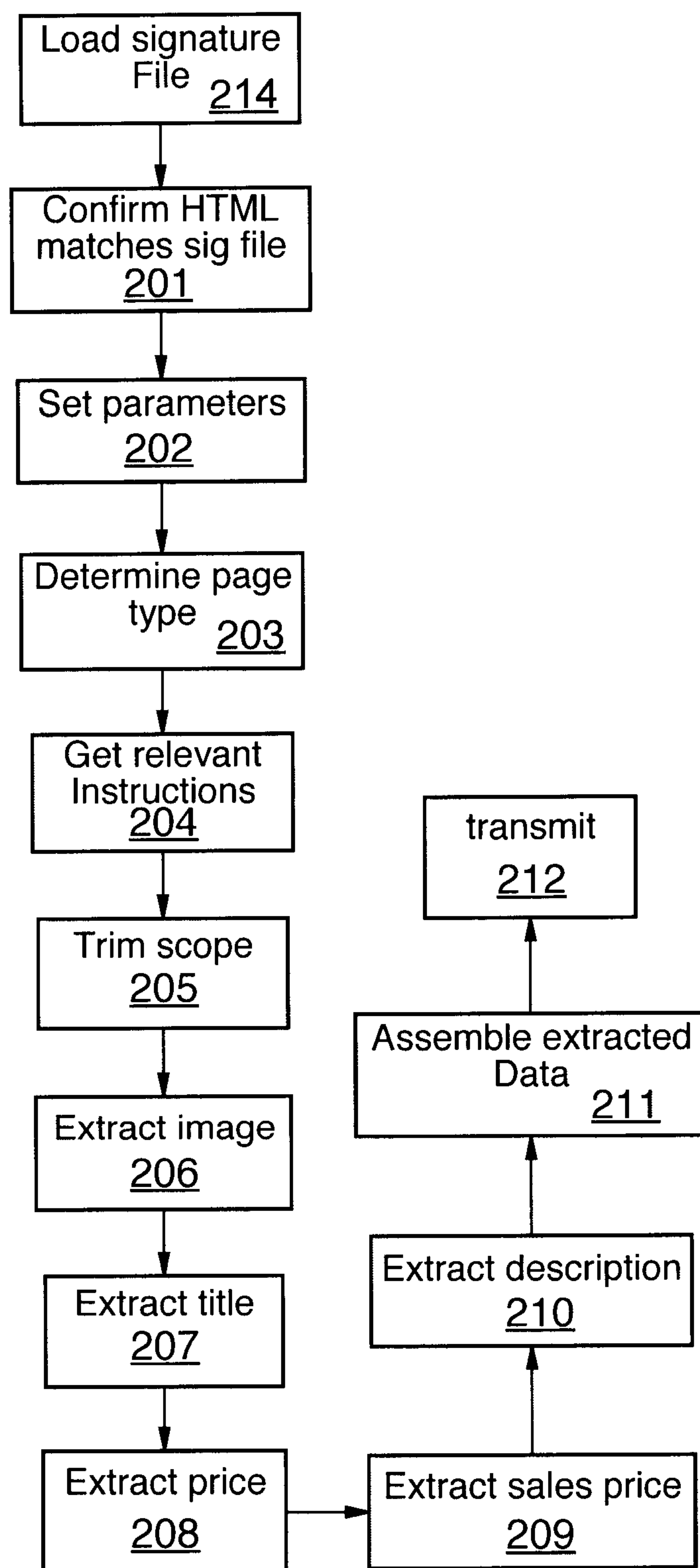


FIG.23

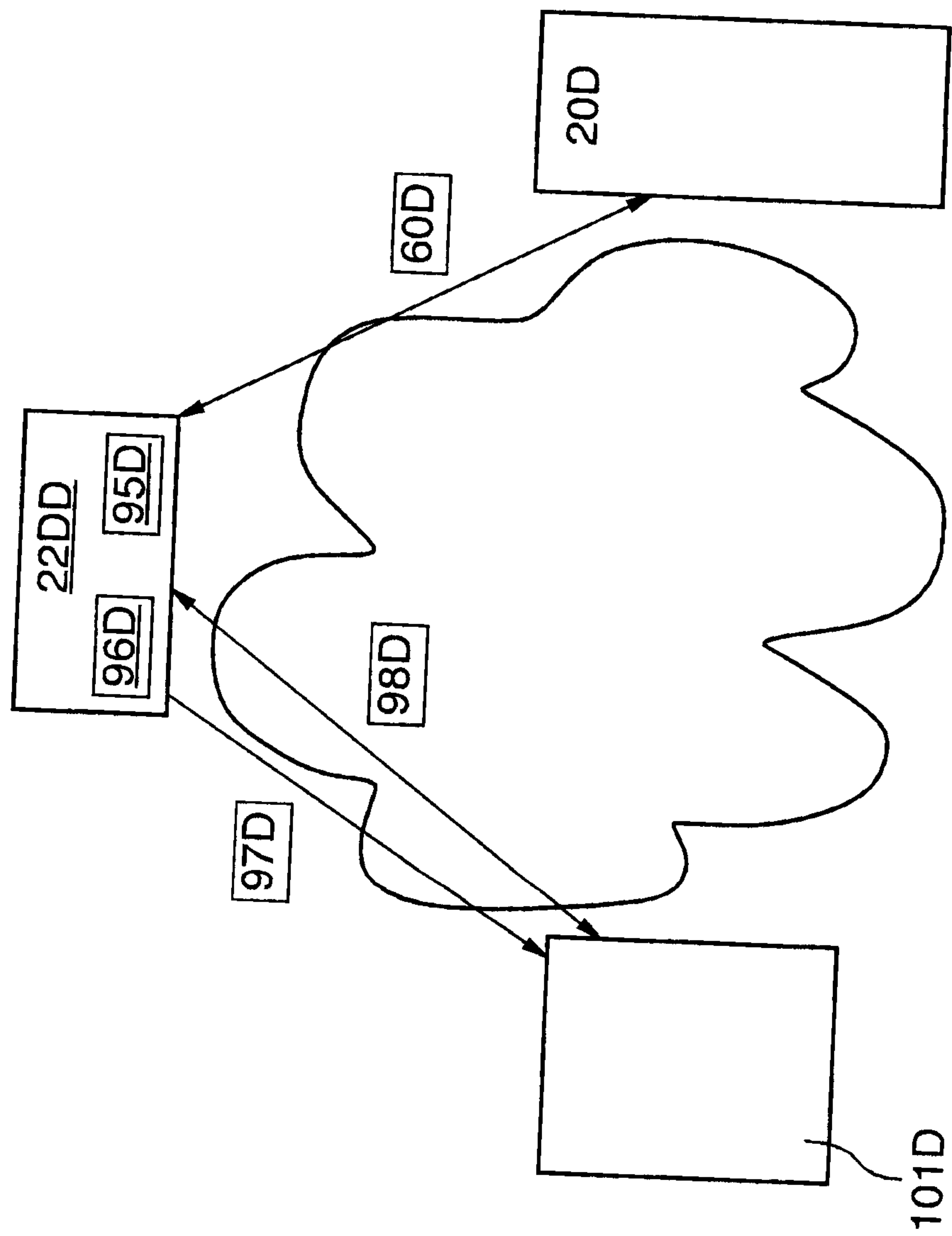


FIG.24

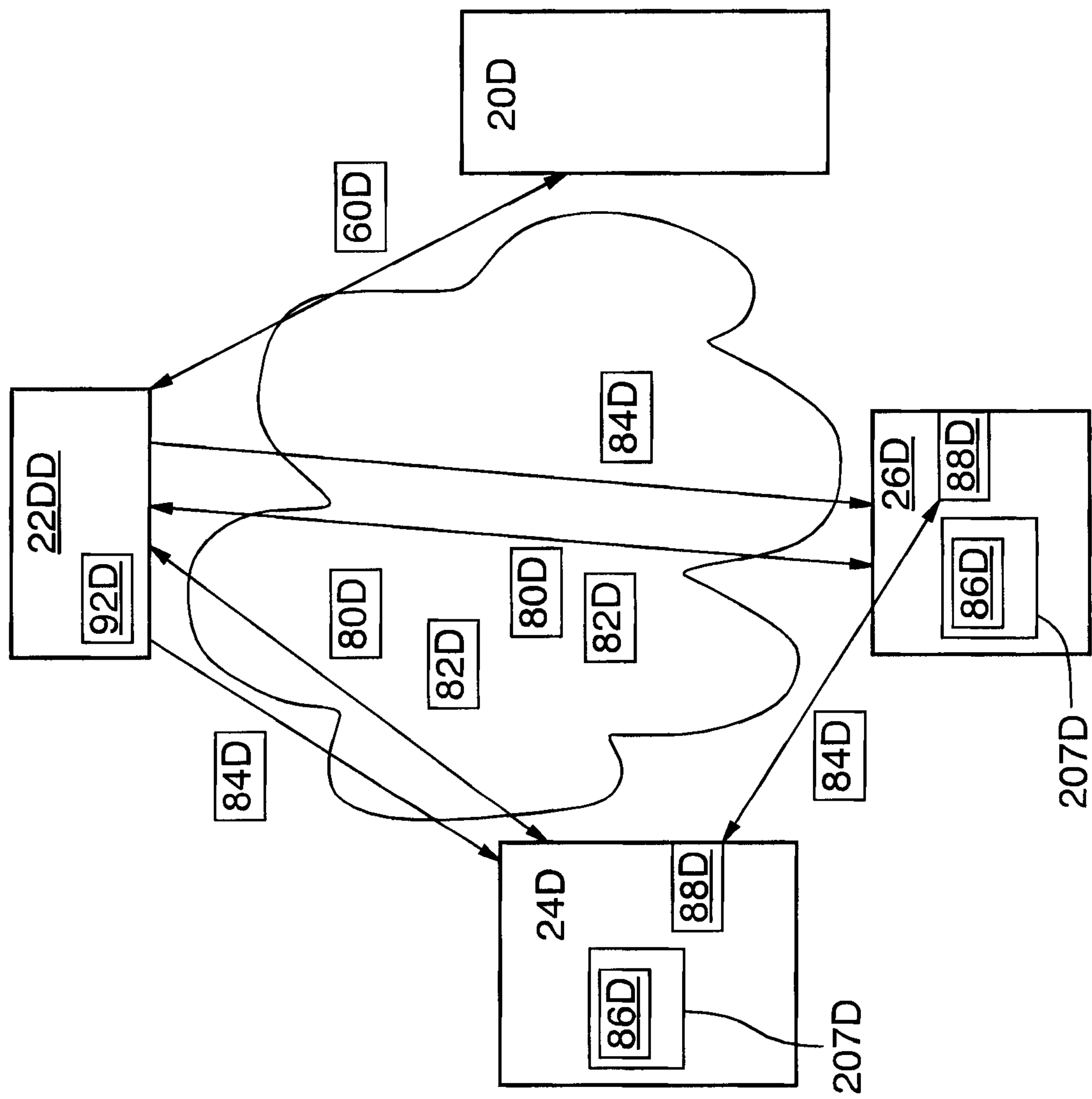


FIG.25

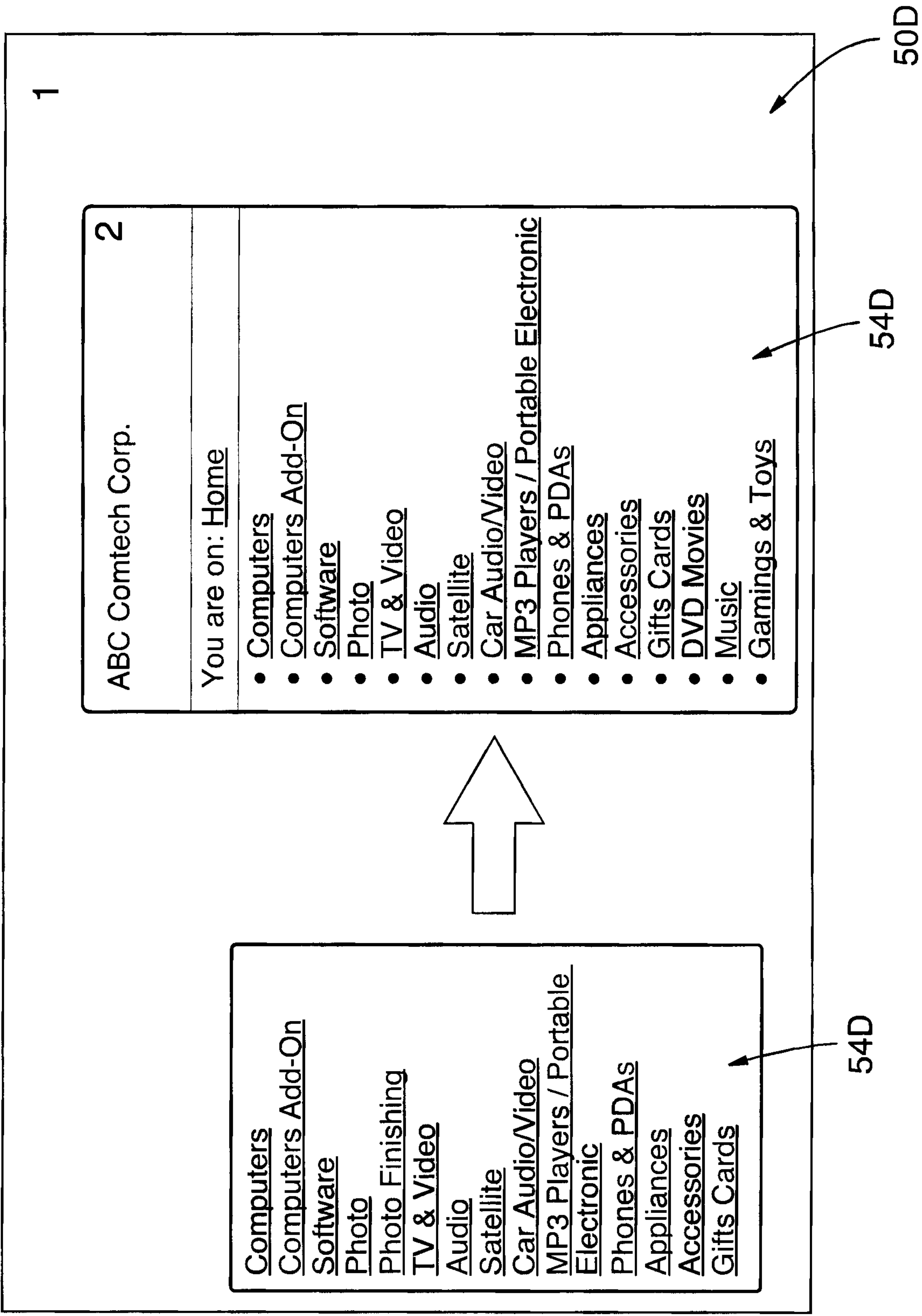


FIG.26

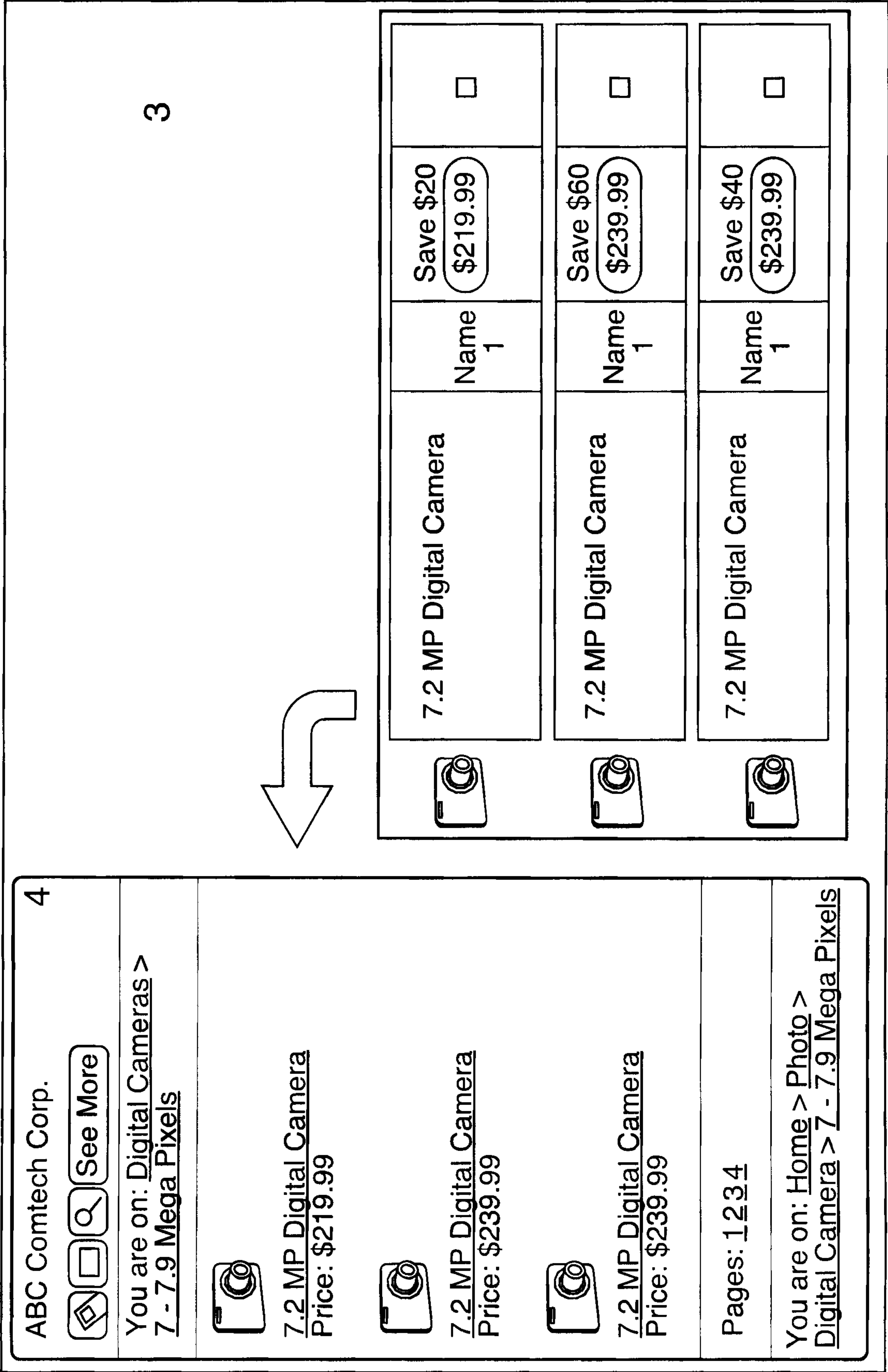


FIG.27A

Table 1

Menu Name	URL
Computers	http://www.ABC Comtech Corp..ca/catalog/dept.asp?logon=&langid=EN&catid=1
Computers Add-Ons	http://www.ABC Comtech Corp..ca/catalog/dept.asp?logon=&langid=EN&catid=18
Software	http://www.ABC Comtech Corp..ca/catalog/dept.asp?logon=&langid=EN&catid=3
Photo Finishing	http://pix.ABC Comtech Corp..ca/en/?Source=FScaNAVbar
Software	http://www.ABC Comtech Corp..ca/catalog/dept.asp?logon=&langid=EN&catid=11

FIG.27B

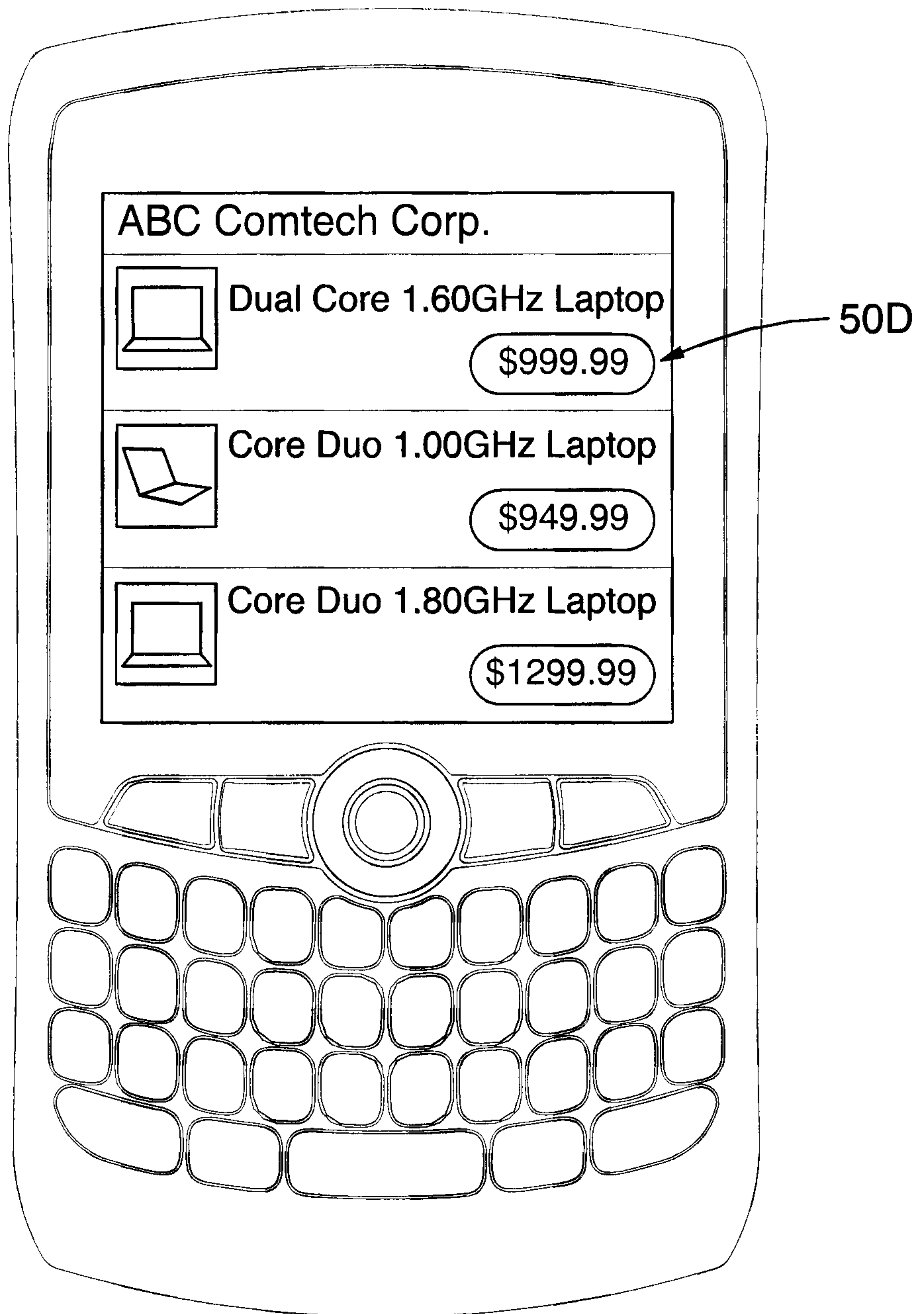


FIG.28

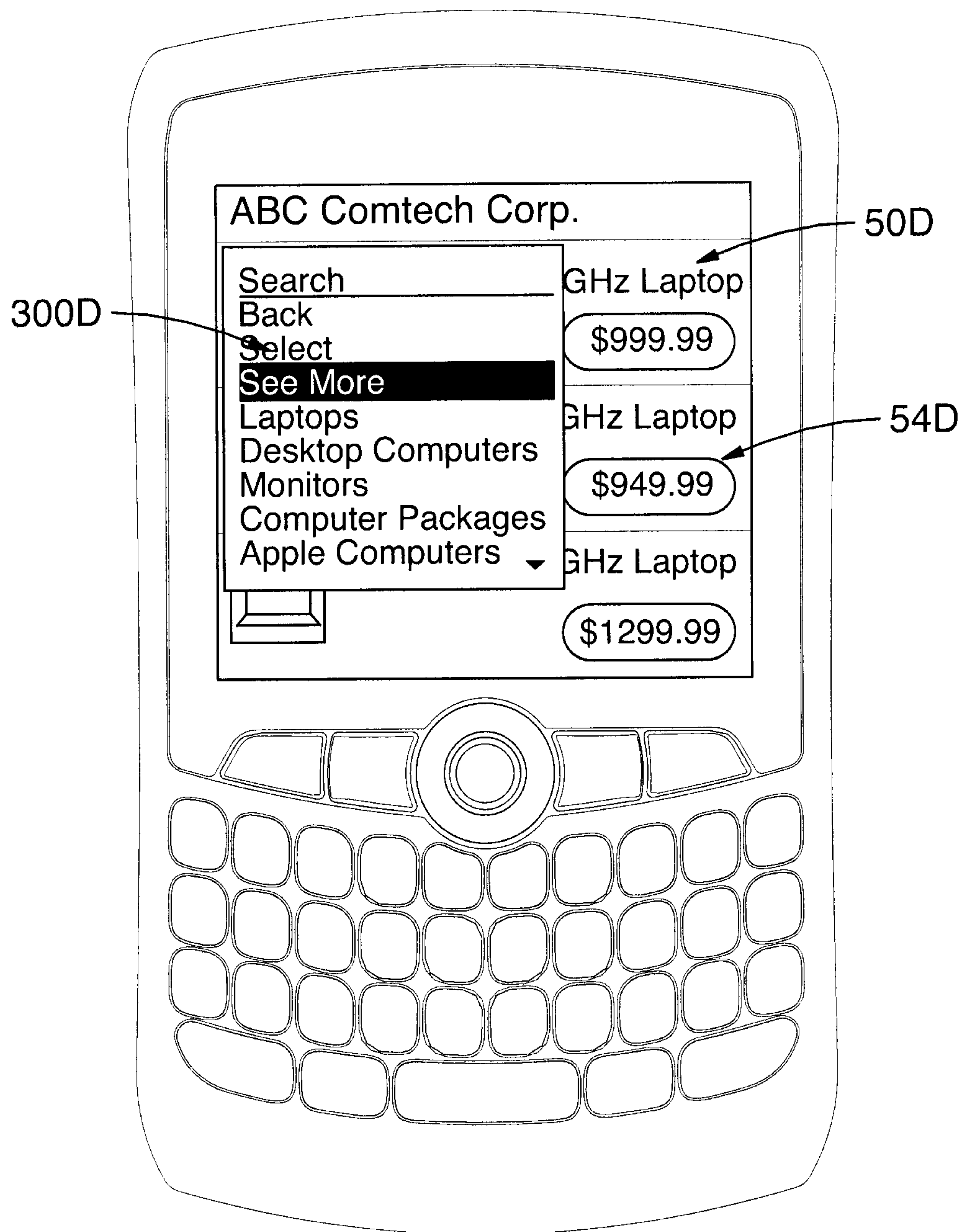


FIG.29

ABC Comtech Corp.

File

Edit

View

Favorites

Tools

Help

Address

http://www.abdcomtech.com

ABC Comtech Corp. Home Page

Shop by department

Brand Showcase

YOU ARE HERE:

Home / Computers / Laptop / 13" Widescreen Laptops

Keyword Search

Entire Site

Computers

► Laptops

13" Widescreen Laptops

Refurbished Laptops

Thin and Light Laptops

Desktop Computers

Monitors

Printers & Fax Machines

Scanners

Save \$50

999.99

Dual Core 1.60GHz Laptop

Our price: \$1049.99

Discount: -\$50.00

Sale Price: \$999.99

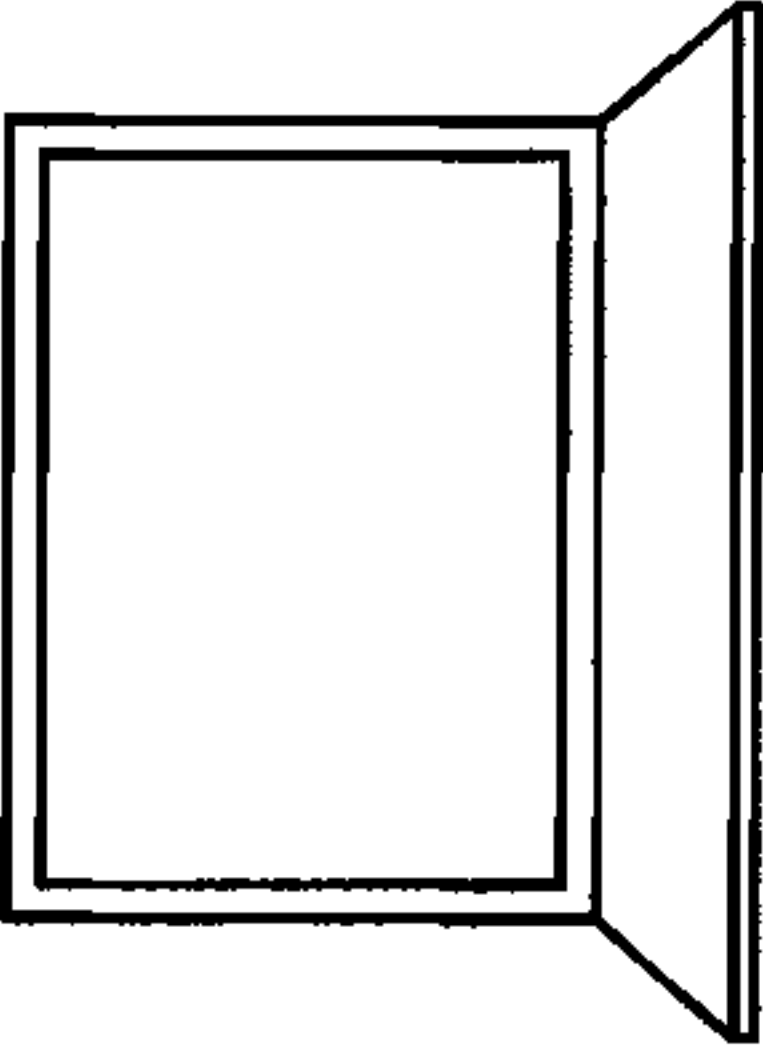


FIG.30

ABC Comtech Corp.

File

Edit

View

Favorites

Tools

Help

Address

http://www.abdcomtech.com

ABC Comtech Corp. Home Page

Shop by department

Brand Showcase

YOU ARE HERE:

Home / TV & Video / 50" & Larger TVs / LCD Flat Panel TVs

Keyword Search

Entire Site

TV & Video

13"-15" TVs

17"-20" TVs

21"-29" TVs

30"-39" TVs

40"-49" TVs

▶ 50" & Larger TVs

LCD Flat Panel TVs

Plasma Flat Panel TVs

DVD Recorders

HD Digital Cable

HD Satellite TV

VCR

>> Compare LCD Flat Panel TVs to all 50" & Larger TVs

1

Sort or compare products

Sort By

COMPARE

1080p 52" Widescreen LCD HDTV

Name 1

\$4259.99

1080p 52" LCD HDTV

Name 2

\$4299.99

1080p 52" Widescreen LCD HDTV

Name 2

\$3959.99

FIG.31

ABC Comtech Corp. File Edit View Favorites Tools Help Address http://www.abdcomtech.com ABC Comtech Corp. Home Page		Shop by department Brand Showcase YOU ARE HERE: Home / Photo / Digital Cameras / 7 - 7.9 Mega Pixels	
Keyword Search Entire Site	Photo Camcorders Bonoculars ▶ Digital Cameras 5 - 5.9 Mega Pixels 6 - 6.9 Mega Pixels 7 - 7.9 Mega Pixels 8 - 8.9 Mega Pixels 9 Mega Pixels and up Digital SLR		
		Save \$40 239.99 7.2 MP Digital Camera - Black Our price: \$279.99 Discount: -\$40.00 Sale Price: \$239.99	

FIG. 32

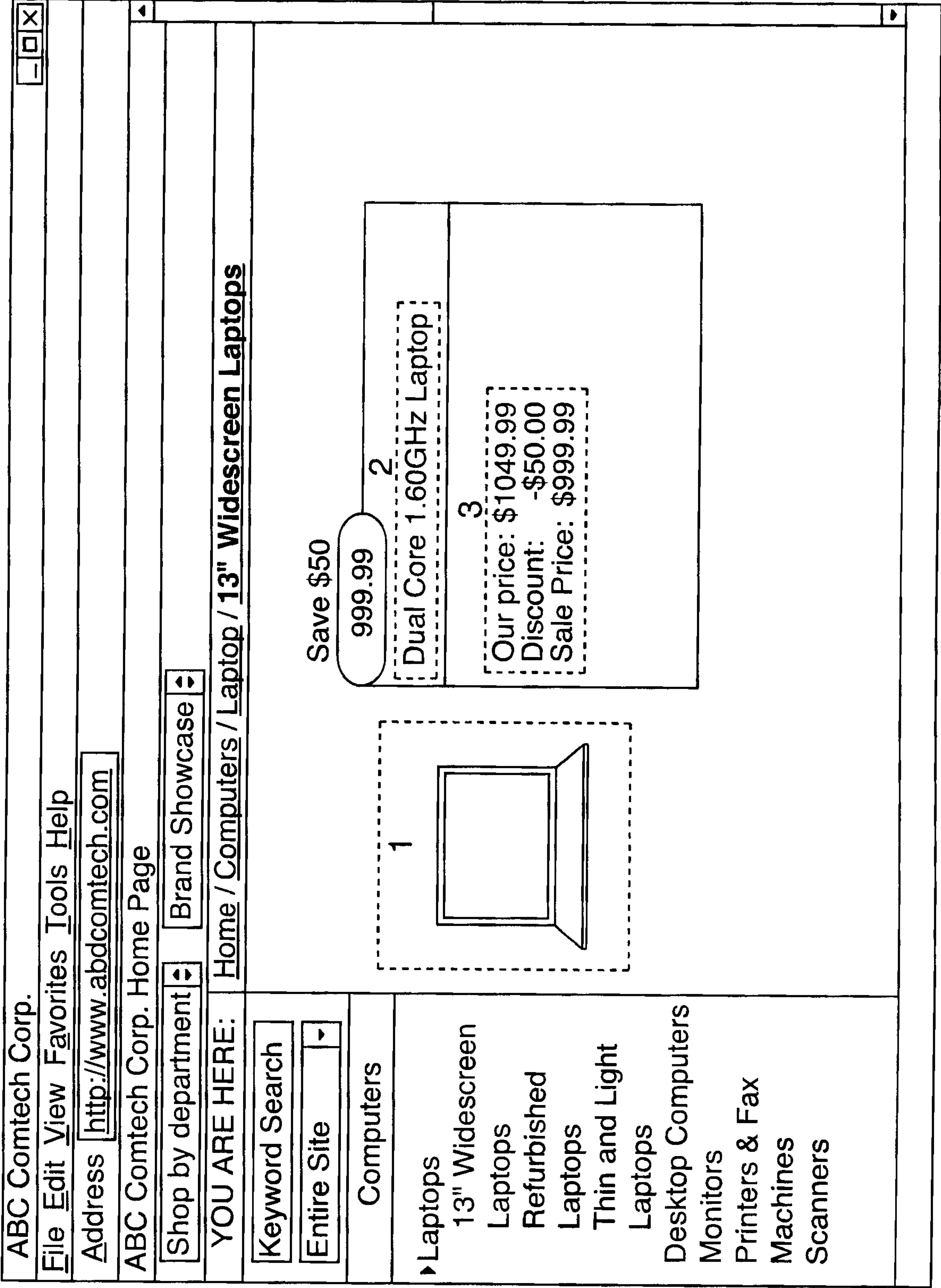


FIG.33

ABC Comtech Corp.

File

Edit

View

Favorites

Tools

Help

Address <http://www.abdcomtech.com>

ABC Comtech Corp. Home Page

Shop by department

Brand Showcase

YOU ARE HERE: [Home](#) / [Photo](#) / [Digital Cameras / 7 - 7.9 Mega Pixels](#)

Keyword Search

Entire Site

Photo

Camcorders

Bonaculares

▶ Digital Cameras

5 - 5.9 Mega Pixels

6 - 6.9 Mega Pixels

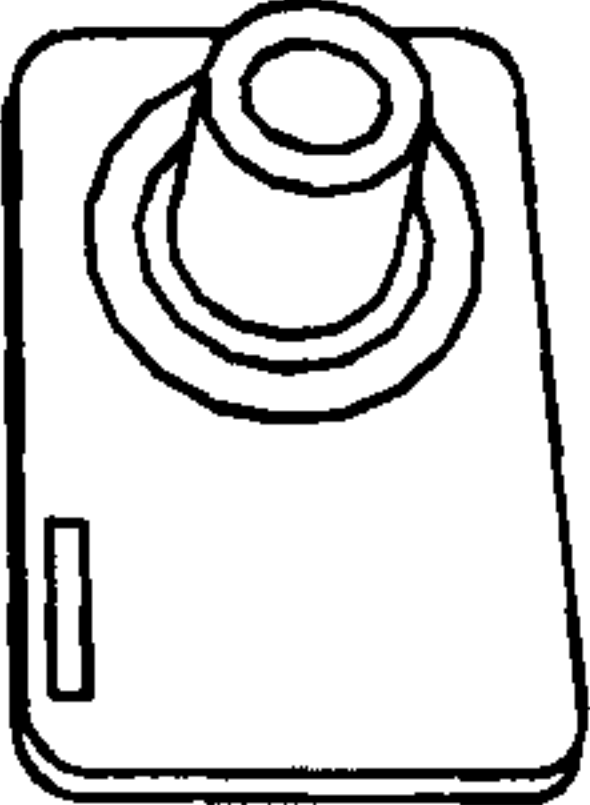
7 - 7.9 Mega Pixels

8 - 8.9 Mega Pixels

9 Mega Pixels and up

Digital SLR

1



2

239.99

3

Our price: \$279.99

Discount: -\$40.00

Sale Price: \$239.99

Save \$40

7.2 MP Digital Camera - Black

FIG.34

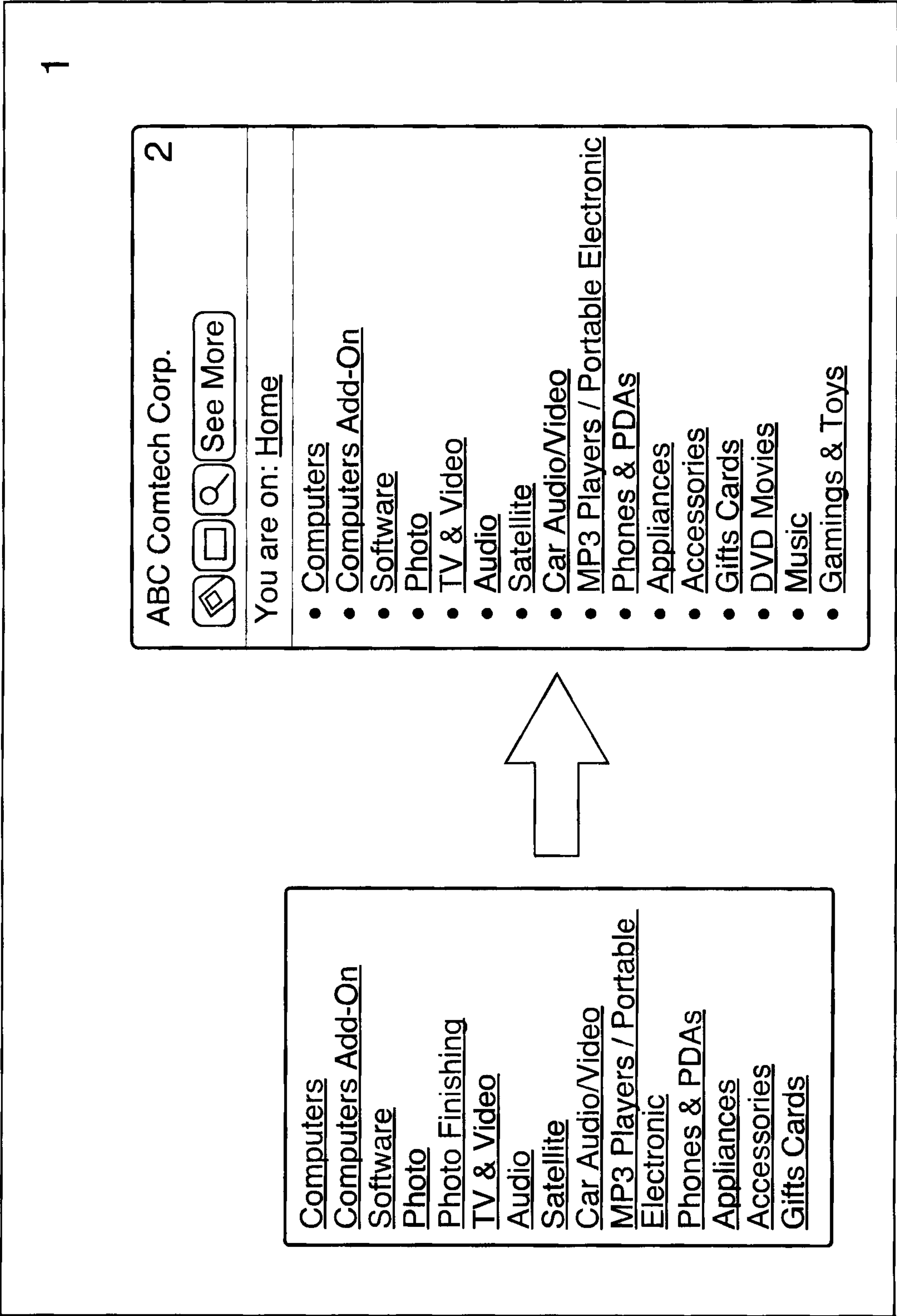


FIG.35

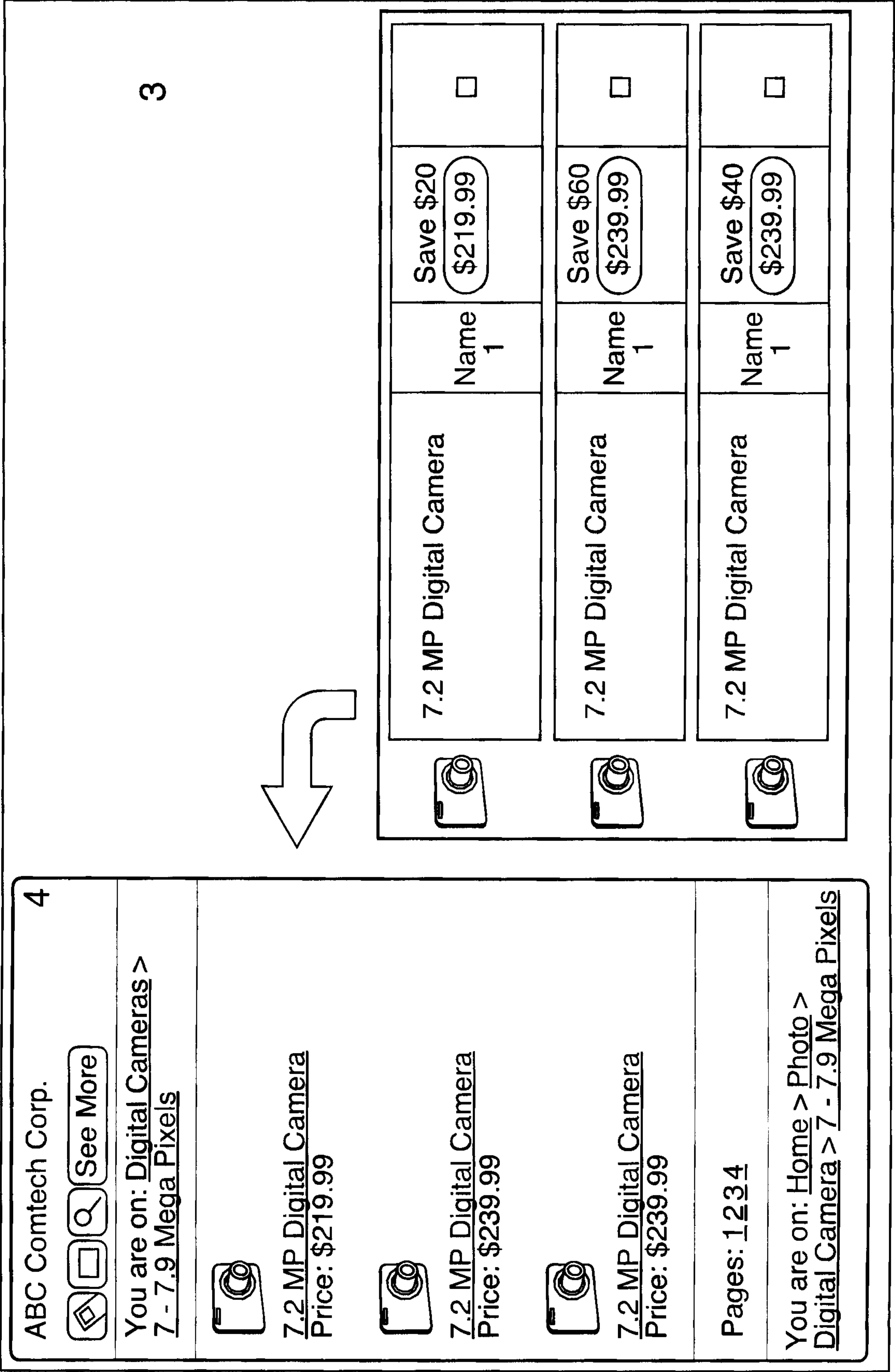


FIG.36

6

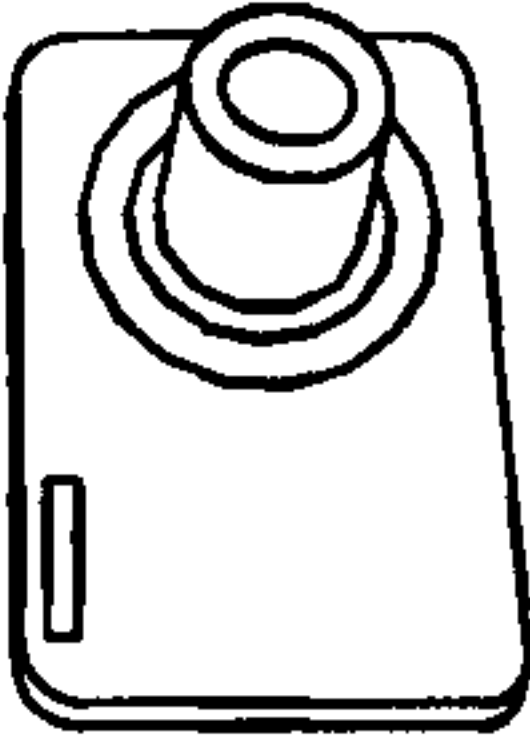
ABC Comtech Corp.



See More

You are on: [7 - 7.9 Mega Pixels](#)

7.2 MP Digital Camera - Black



[Save item](#) | [Purchase item](#)

Price: \$279.99

Sale Price: \$239.99

Description:
The slim, compact sports model 7.2-megapixel imager and precision lenses. With a 2.5-inch LCD screen wrapped in a metal body. This model features 3x optical zoom capability, blur-reduction technology, and substantial internal memory.

You are on: [Home](#) > [Photo](#) > [Digital Camera](#) > [7 - 7.9 Mega Pixels](#)

5

Save \$40

239.99

7.2 MP Digital Camera - Black

Our price: \$279.99

Discount: -\$40.00

Sale Price: \$239.99

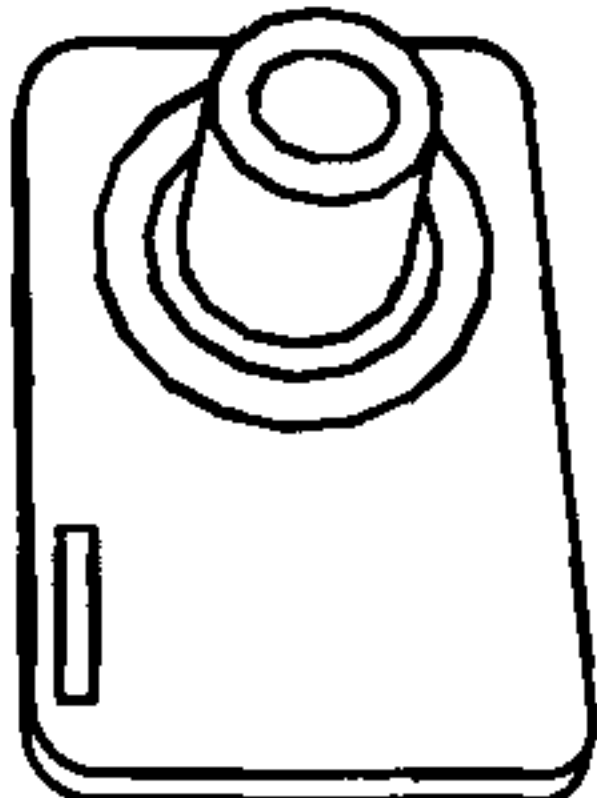


FIG.37

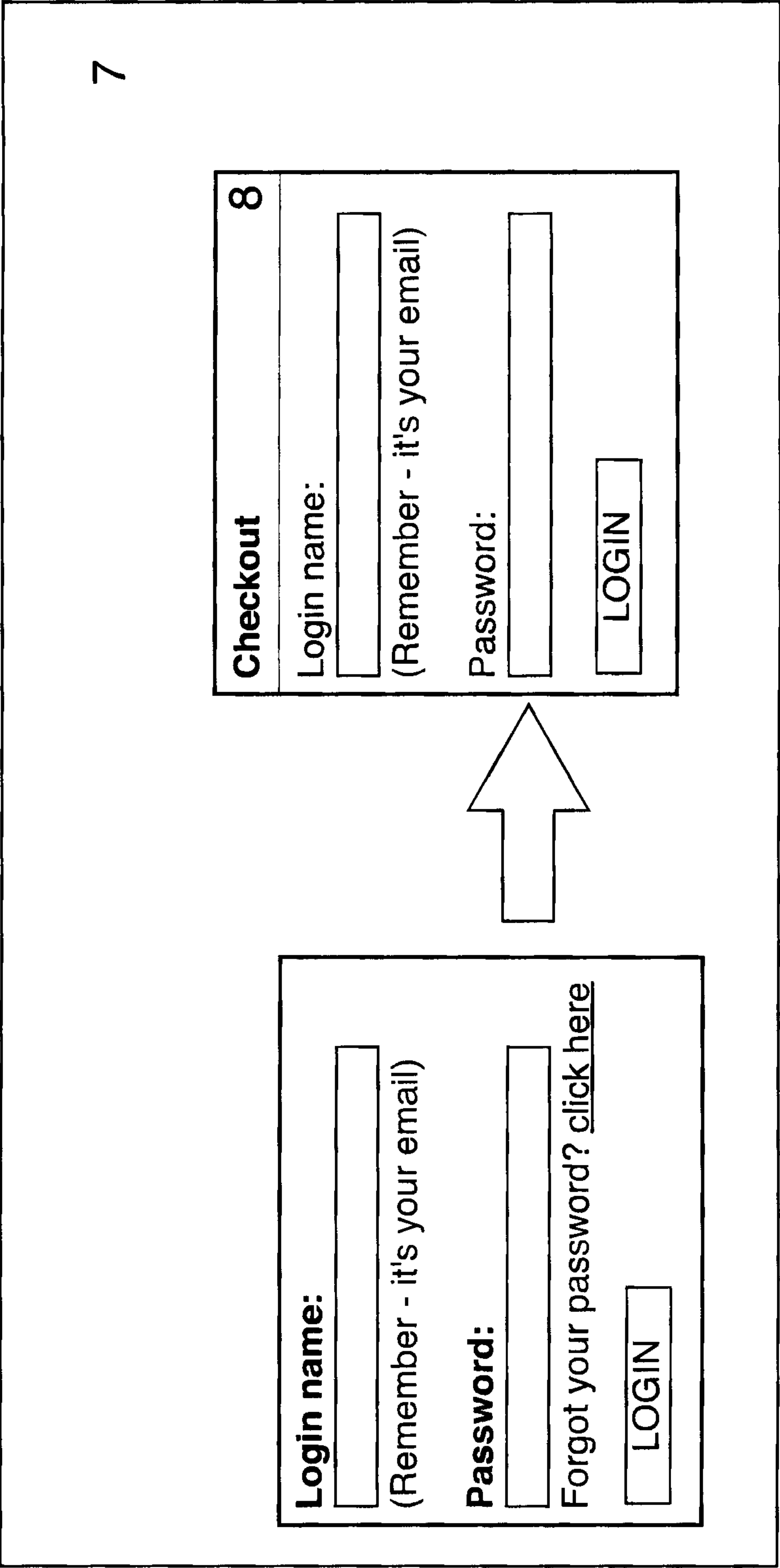


FIG.38

9	ABC Comtech Corp.	10
Checkout		
Order Summary		
Product Total:	\$219.99	
Shipping:	\$7.99	
Provincial Sales Tax (PST):	\$19.64	
Goods and Services Tax (GST):	\$14.68	
Total:	\$282.70	
Credit Card Payment		
Type	Visa	
Credit Card Number		
Expiry Date	Mo	Year
Card Holder First Name		
Card Holder Last Name		

Order Summary

Product Total: \$219.99

Shipping: \$7.99

Provincial Sales Tax (PST): \$19.64

Goods and Services Tax (GST): \$14.68

Total: \$282.70

Type *
Visa

Credit Card Number *

Expiry Date - (Not required for Store credit cards)

Mo Year

Card Holder First Name *
Card Holder Last Name *



FIG.39

ABC Comtech Corp.

X

File

E

dit

V

iew

F

avorites

T

ools

H

elp

ABC Comtech Corp. Home Page

SEARCH FOR

Keyword or Item #

 IN

All Categories

GO

TELEVISIONS

By TV Type

Plasma Flat Panel

LCD Flat Panel

Projection

Tube

Combo

Handheld

Front Projections & Screen

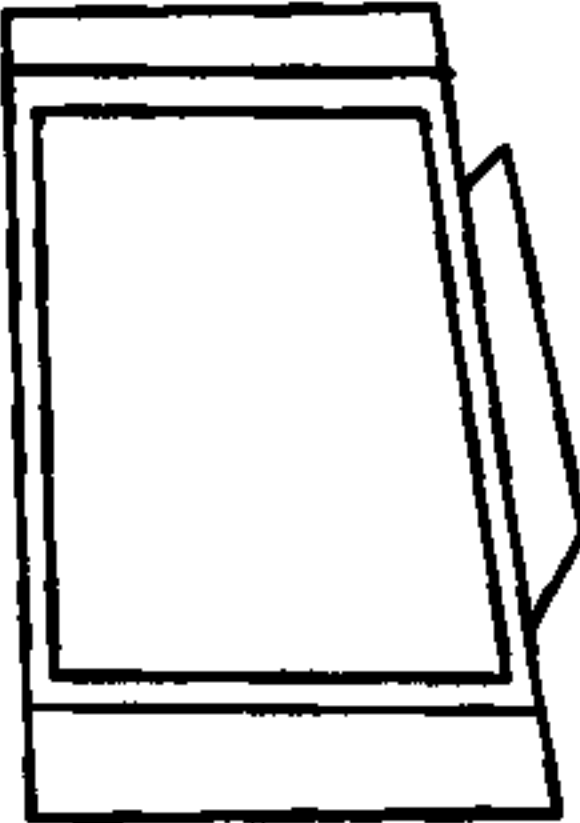
By Price Range

By Size

By Brand

By Premium TVs

Home > Television > By TV Type > Plasma Flat Panel > Product Info



42" Plasma HDTV
Model: PL4260M

Watch all of your favorite high-definition quality broadcasts on this 42" plasma TV.

VIEW MORE PHOTOS

Our Price \$ 1 199.99

- Built-In HDTV Tuner: Add HD-capable antenna to receive over-the-air high-definition broadcasts, where available.
- 1024 x 768 pixel resolution

FIG.40

ABC Comtech Corp.
File Edit View Favorites Tools Help

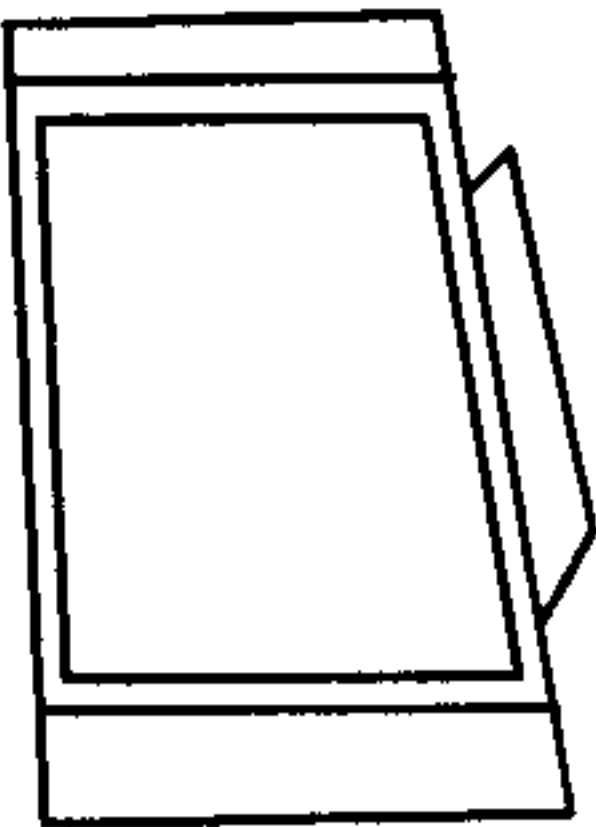
Fat Free Mobile
larry.simsom

My Home

Contact Us

☒

How does this work?



Title: 42" Plasma HDTV

Desc Watch all of your favorite high-definition quality broadcasts on this 42" plasma TV.

Price Our Price: \$199.99 USD

Save Now

☐ Stay here!

Cancel

97

ABC Comtech Corp. Home Page

SEARCH FOR IN All Categories

TELEVISIONS

By TV Type

Plasma Flat Panel

LCD Flat Panel

Projection

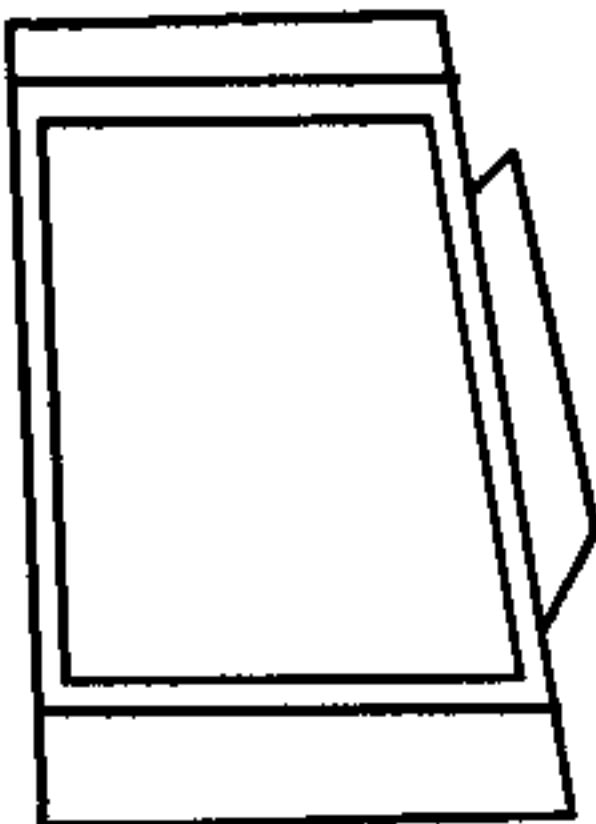
Tube

Combo

Handheld

Front Projections &

Home > Television > By TV Type > Plasma Flat Panel > Product Info



42" Plasma HDTV
Model: PL4260M

Watch all of your favorite high-definition quality broadcasts on this 42" plasma TV.

VIEW MORE PHOTOS

Our Price \$ 1199.99

• Built-In HDTV Tuner: Add HD-capable antenna

FIG.41

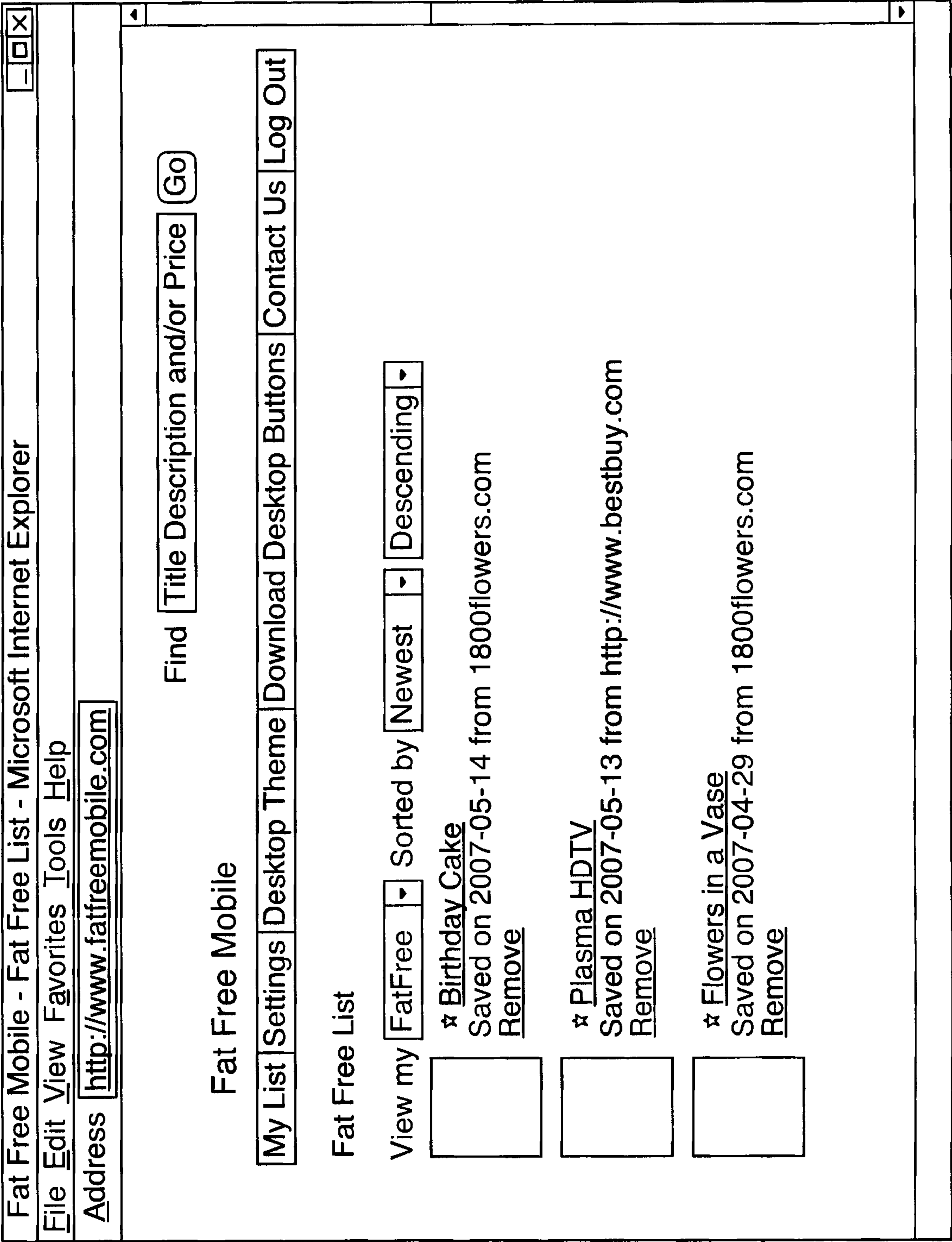


FIG.42

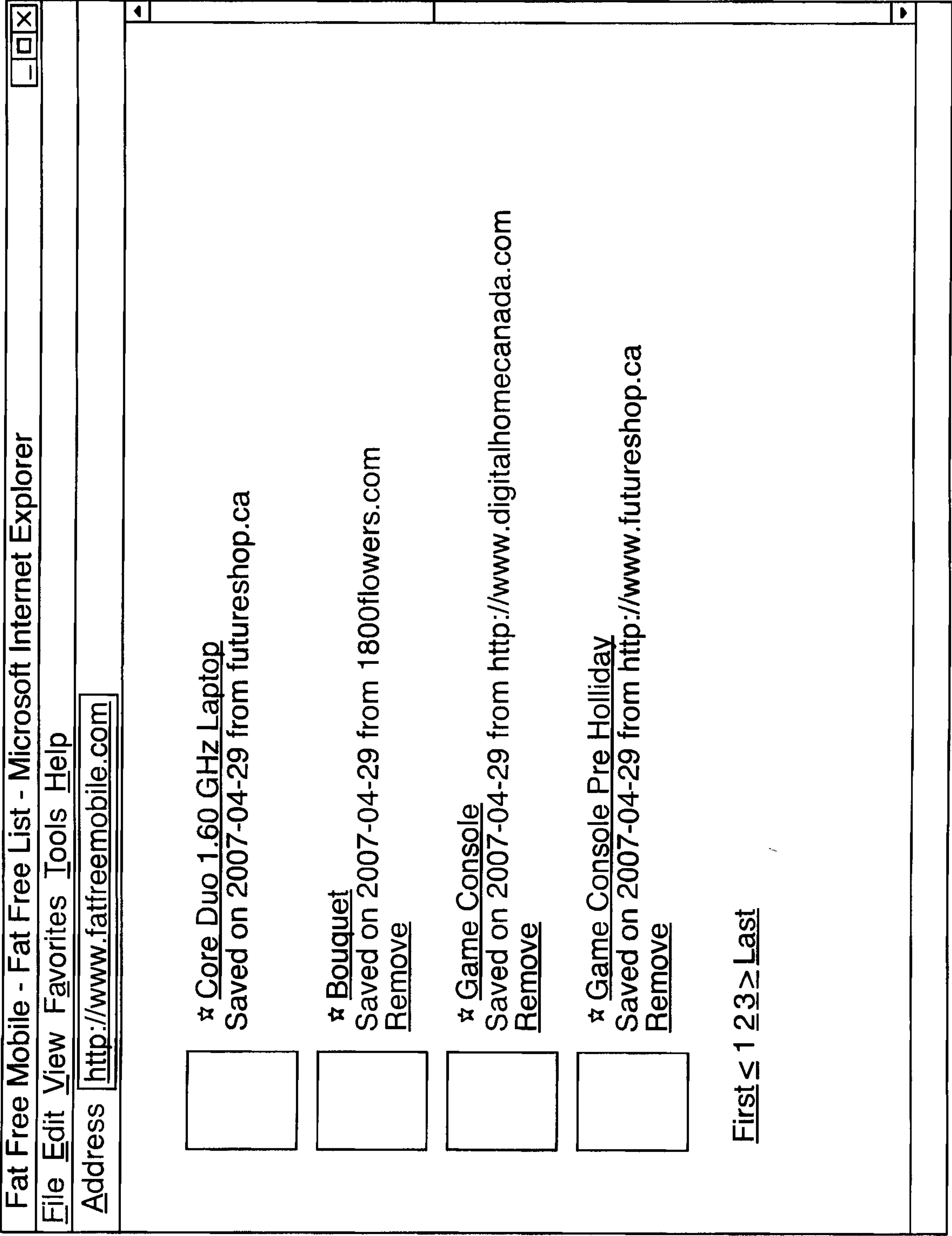


FIG.43

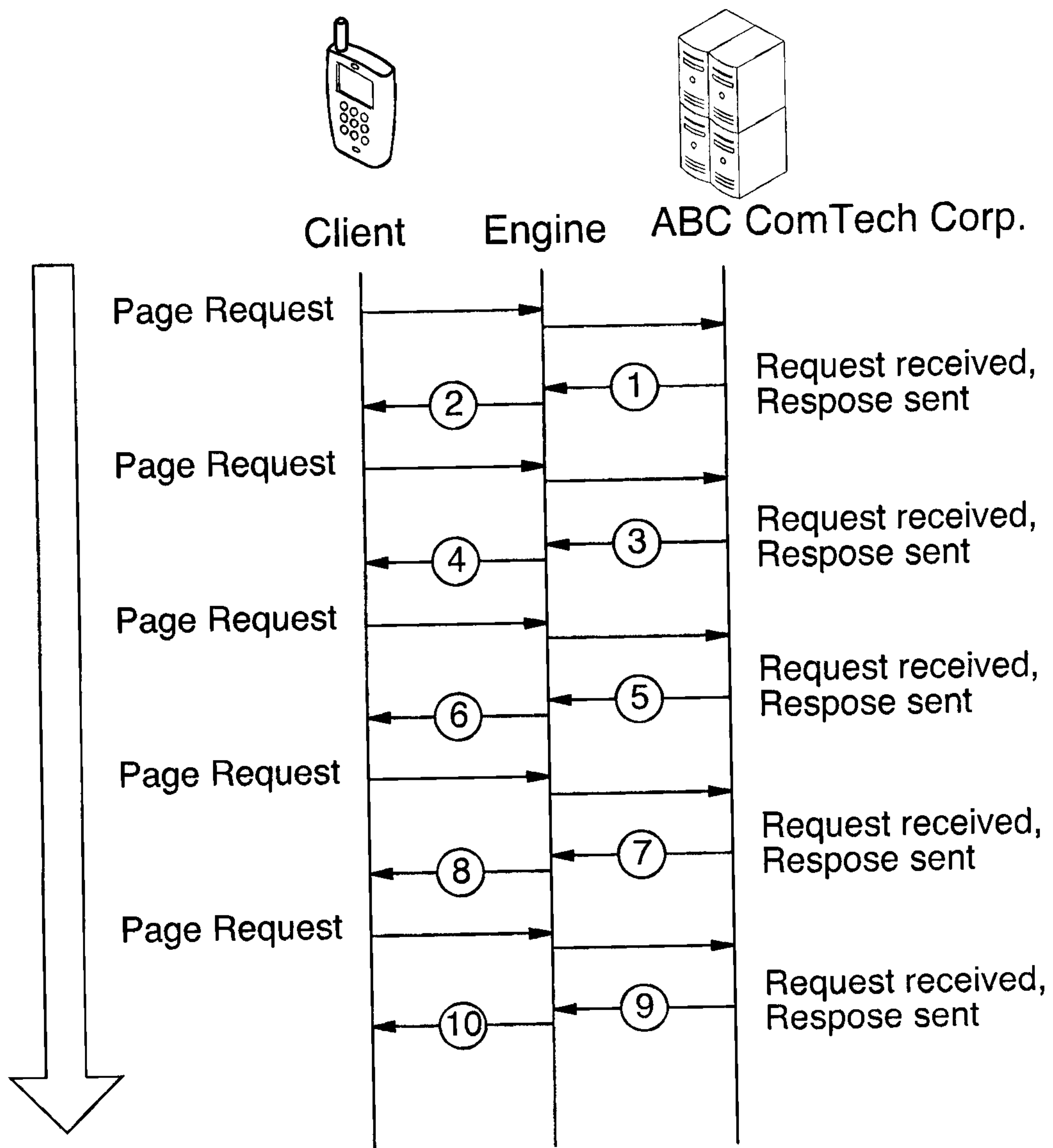


FIG.44

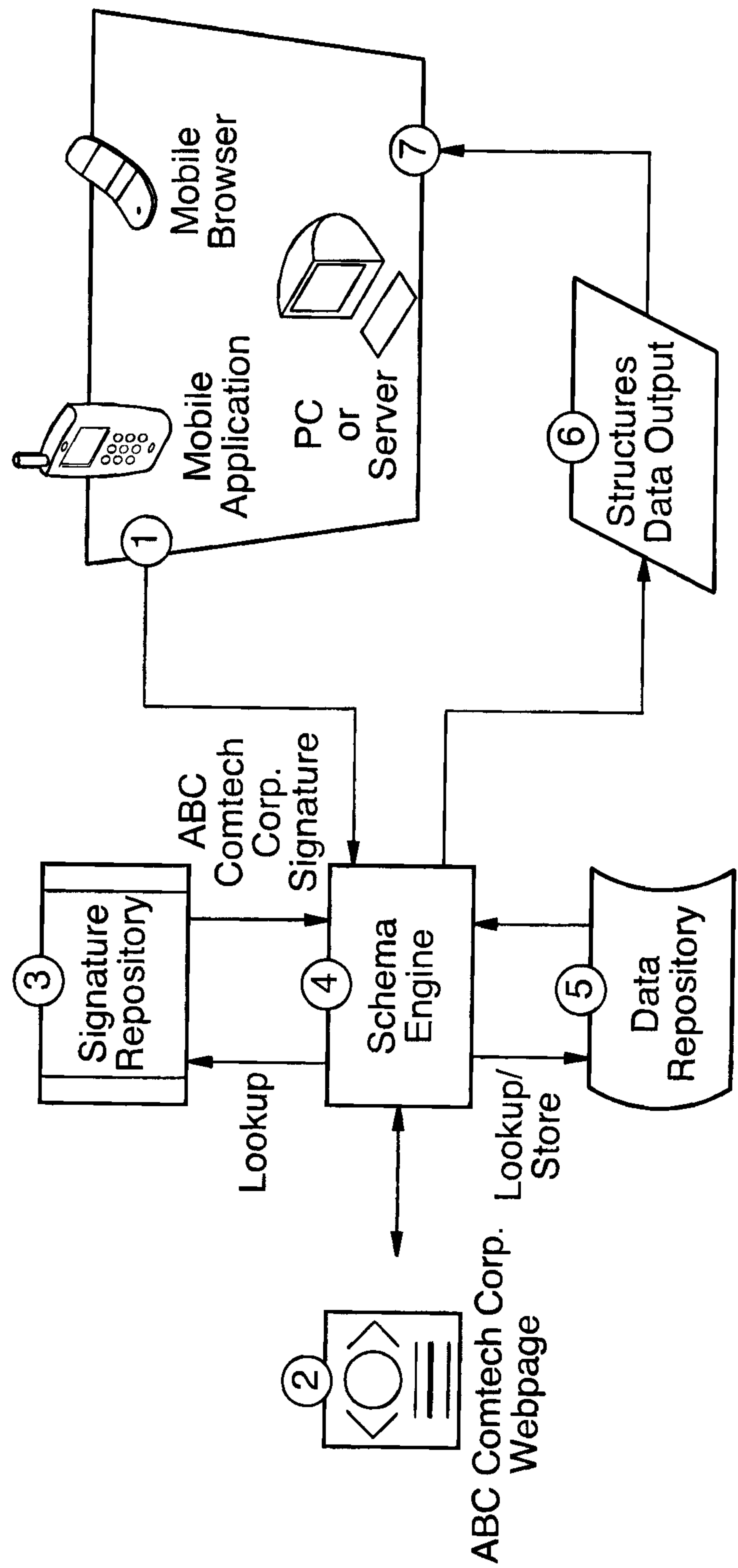


FIG.45