(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 188 113 B1

(12) **EUROPEAN PATENT SPECIFICATION**

(45) Date of publication and mention of the grant of the patent:
05.10.2005 Bulletin 2005/40

(21) Application number: **00912127.8**

(22) Date of filing: **29.02.2000**

(51) Int Cl.[7]: **G06F 9/355**, G06F 9/318, G06F 12/02

(86) International application number:
PCT/US2000/005420

(87) International publication number:
WO 2000/055723 (21.09.2000 Gazette 2000/38)

(54) **LINEAR ADDRESS EXTENSION AND MAPPING TO PHYSICAL MEMORY USING 4 AND 8 BYTE PAGE TABLE ENTRIES IN A 32-BIT MICROPROCESSOR**

ERWEITERUNG UND ABBILDUNG AUF PHYSIKALISCHEM SPEICHER VON LINEAREN ADRESSEN UNTER VERWENDUNG VON 4 UND 8 BYTE SEITENTABELLENEINTRÄGEN IN EINEM 32-BIT MIKROPROZESSOR

EXTENSION D'ADRESSE LINEAIRE ET MAPPAGE DANS UNE MEMOIRE PHYSIQUE UTILISANT DES ENTREES DE TABLEAU DE PAGE 4 ET 8 OCTETS DANS UN MICROPROCESSEUR 32 BITS

(84) Designated Contracting States:
DE FR GB

(30) Priority: **12.03.1999 US 267796**

(43) Date of publication of application:
20.03.2002 Bulletin 2002/12

(73) Proprietor: **INTEL CORPORATION**
Santa Clara, CA 95052 (US)

(72) Inventors:
- **SHAHIDZADEH, Shahrokh**
  Beaverton, OR 97007 (US)
- **BIGBEE, Bryant, E.**
  Aloha, OR 97006 (US)
- **PAPWORTH, David, B.**
  Beaverton, OR 97007 (US)
- **BINNS, Frank**
  Hillsboro, OR 97123 (US)
- **COLWELL, Robert, P.**
  Portland, OR 97229 (US)

(74) Representative: **Molyneaux, Martyn William et al**
Harrison Goddard Foote
40-43 Chancery Lane
London WC2A 1JA (GB)

(56) References cited:
EP-A- 0 530 682 **US-A- 4 679 140**
US-A- 5 566 308

- **C RAY PENG ET AL: "THE POWER ARCHITECTURETM: 64-BIT POWER WITH 32-BIT COMPATIBILITY" DIGEST OF PAPERS OF THE COMPUTER SOCIETY COMPUTER CONFERENCE (SPRING) COMPCON,US,LOS ALAMITOS, IEEE COMP. SOC. PRESS, vol. CONF. 40, 1995, pages 300-307, XP000545443 ISBN: 0-7803-2657-1**

EP 1 188 113 B1

Description

Field

[0001] The present invention relates to microprocessor and computer systems, and more particularly, to virtual memory systems with extended linear address generation and translation.

Background

[0002] Most microprocessors make use of virtual or demand-paged memory schemes, where sections of a program's execution environment are mapped into physical memory as needed. Virtual memory schemes allow the use of physical memory much smaller in size than the linear address space of the microprocessor, and also provide a mechanism for memory protection so that multiple tasks (programs) sharing the same physical memory do not adversely interfere with each other.

[0003] Physical memory is part of a memory hierarchy system, which may be illustrated as part of a computer system shown in Fig. 1. Microprocessor **102** has a first level cache comprising instruction cache **104** and data cache **106.** Microprocessor **102** communicates with unified second level cache **108** via backside bus **110.** Second level cache **108** contains both instructions and data, and may physically reside on the chip die **102.** Caches **104** and **106** comprise the first level of the memory hierarchy, and cache **108** comprises the second level.

[0004] The third level of memory hierarchy for the exemplary computer system of Fig. 1. is indicated by memory **112.** Microprocessor **102** communicates with memory **112** via host processor (front side) bus **114** and chipset **116.** Chipset **116** may also provide graphics bus **118** for communication with graphics processor **120,** and serves as a bridge to other busses, such as peripheral component bus **122.** Secondary storage, such as disk unit **124,** provides yet another level in the memory hierarchy.

[0005] Fig. 2 illustrates some of the functional units within microprocessor **102,** including the instruction and data caches. In microprocessor **102,** fetch unit **202** fetches instructions from instruction cache **104,** and decode unit **206** decodes these instructions. For a CISC (Complex Instruction Set Computer) architecture, decode unit 206 decodes a complex instruction into one or more micro-instructions. Usually, these microinstructions define a load-store type architecture, so that micro-instructions involving memory operations are simple load or store operations. However, the present invention may be practiced for other architectures, such as for example RISC (Reduced Instruction Set Computer) or VLIW (Very Large Instruction Word) architectures.

[0006] For a RISC architecture, instructions are not decoded into micro-instructions. Because the present invention may be practiced for RISC architectures as well as CISC architectures, we shall not make a distinction between instructions and micro-instructions unless otherwise stated, and will simply refer to these as instructions.

[0007] Most instructions operate on several source operands and generate results. They name, either explicitly or through an indirection, the source and destination locations where values are read from or written to. A name may be either a logical (architectural) register or a location in memory. Renaming logical registers as physical registers may allow instructions to be executed out of order. In Fig. 2, register renaming is performed by renamer unit **208,** where RAT (Register Allocation Table) **210** stores current mappings between logical registers and physical registers. The physical registers are indicated by register file **212.**

[0008] Every logical register has a mapping to a physical register in physical register file **212,** where the mapping is stored in RAT **210** as an entry. An entry in RAT **210** is indexed by a logical register and contains a pointer to a physical register in physical register file **212.** Some registers in physical register file **212** may be dedicated for integers whereas others may be dedicated for floating point numbers, but for simplicity these distinctions are not indicated in Fig. 2.

[0009] During renaming of an instruction, the current RAT provides the required mapping for renaming the source logical register(s) of the instruction, and a new mapping is created for the destination logical register of the instruction. This new mapping evicts the old mapping in the RAT.

[0010] Renamed instructions are placed in instruction window buffer **216.** All instructions "in-flight" have an entry in instruction window buffer **216,** which operates as a circular buffer. Instruction window buffer **216** allows for memory disambiguation so that memory references are made correctly, and allows for instruction retirement in original program order. (For CISC architectures, a complex instruction is retired when all micro-instructions making up the complex instruction are retired together.)

[0011] For an instruction that writes its result to a memory location, data cache 106 (part of the memory hierarchy) is updated upon instruction retirement. For an instruction that writes its result to a logical register, no write need be done upon retirement because there are no registers dedicated as logical registers. (Physical register file **212** has the result of the retiring instruction in that physical register which the destination logical register was mapped to when the instruction was renamed.)

[0012] Scheduler **218** schedules instructions to execution units **220** for execution. For simplicity, only memory execution unit **224** is explicitly indicated in execution units **220.** A load or store instruction is dispatched by scheduler **218** to AGU (Address Generation Unit) **222** for computation of a linear address, and memory execution unit **224** translates the linear address into a physical address and executes the load or store instruction. Memory execution unit may send data to or receive data

from a forwarding buffer (not shown) rather than data cache **106,** where a forwarding buffer stores objects that may eventually be written to data cache **106** upon instruction retirement. The scheduling function performed by scheduler **218** may, for example, be realized by reservation stations (not shown) implementing Tomasulo's algorithm (or variations thereof) or by a scoreboard. Execution units **220** may retrieve data from or send data to register file **212,** depending upon the instruction to be executed.

[0013] In other embodiments of the present invention, the information content contained in the data structures of physical register field **212** and instruction window buffer **216** may be realized by different functional units. For example, a re-order buffer may replace instruction window buffer **216** and physical register file **212,** so that results are stored in the re-order buffer, and in addition, registers in a register file are dedicated as logical registers. For this type of embodiment, the result of an instruction that writes to a logical register is written to a logical register upon instruction retirement.

[0014] With most modem computer systems, a microprocessor refers to a memory location by generating a linear address, but an object is retrieved from a specific memory location by providing its physical address on an address bus, such as bus **114** in Fig. 1. Linear addresses may be the same as physical addresses, in which case address translation is not required. However, usually a virtual memory scheme is employed in which linear addresses are translated into physical addresses. In this case, a linear address may also be referred to as a virtual address. The linear address space is the set of all linear addresses generated by a microprocessor, whereas the physical address space is the set of all physical addresses.

[0015] For some microprocessor architectures, such as Intel@ Architecture 32 bit (IA-32) microprocessors (Intel® is a registered trademark of Intel Corporation, Santa Clara, California), there is also another type of address translation in which a logical address is translated into a linear address. For these type of architectures, the instructions provide logical address offsets, which are then translated to linear addresses by AGU **222** in Fig. 2. This extra stage of address translation may provide additional security, e.g., where application code cannot modify supervisory (operating system) code.

[0016] The mapping of a logical address to a linear address is illustrated in Fig. 3. A logical address comprises segment selector **302a** and offset **304.** Segment selector **302a** is stored in segment register **302,** which also contains descriptor cache **302b.** Segment selector **302a** points to segment descriptor **308** in descriptor table **306.** Descriptor table **306** provides a table of segment descriptors stored in memory. A segment descriptor provides a segment base address, so that a linear address is obtained by adding an offset to the base address provided by a segment descriptor, as indicated by summation **312.** In addition to providing a base address, a segment descriptor contains various other types of information, such as access rights and segment size. The base address, access rights, segment size, and other information, is cached in descriptor cache **302b.**

[0017] A virtual or demand-paged memory system may be illustrated as a mapping between a linear (virtual) address space and a physical address space, as shown in Fig. 4. In a virtual memory system, the linear and physical address spaces are divided into blocks of contiguous addresses, customarily referred to as pages if they are of constant size or are any of several fixed sizes. A typical page size may be 4KBytes, for example.

[0018] The mapping shown in Fig. 4 illustrates a generic two-level hierarchical mapping comprising directory tables and page tables. Page directory tables and page tables are stored in physical memory, and are usually themselves equal in size to a page. A page directory table entry (PDE) points to a page table in physical memory, and a page table entry (PTE) points to a page in physical memory. For the two-level hierarchical mapping of Fig. 4, a linear address comprises directory field **402,** table field **404,** and offset field **406.** A directory field is an offset to a PDE, a table field is an offset to a PTE, and an offset field is an offset to a memory location in a page.

[0019] In Fig. 4, page directory base register (PDBR) **408** points to the base address of page directory **410,** and the value stored in directory field **402** is added to the value stored in PDBR **408** to provide the physical address of PDE **412** in page directory **410.** PDE 412 in turn points to the base address of page table **414,** which is added to the value stored in table field **404** to point to PTE **416** in page table **414.** PTE **416** points to the base address of page **418,** and this page base address is added to the value stored in offset **406** to provide physical address **420.** Linear address **422** is thereby mapped to physical address **420.**

[0020] Accessing entries stored in page directories and page tables require memory bus transactions, which can be costly in terms of processor cycle time. However, because of the principle of locality, the number of memory bus transactions may be reduced by storing recent mappings between linear and physical addresses in a cache, called a translation look-aside buffer (TLB). There may be separate TLBs for instruction addresses and data addresses. Entries in a TLB are indexed by linear addresses. A hit in a TLB provides the physical address associated with a linear address. If there is a miss, then the memory hierarchy is accessed sometimes referred to as a page walk as indicated in Fig. 4 to obtain the translation of a linear address into a physical address.

[0021] Some IA-32 microprocessors employ several modes for translating linear addresses into physical addresses, and we shall consider three such modes herein referred to as modes A, B, and C. Mode A supports a 32 bit physical address space with 4KB page sizes. Mode B supports a 32 bit physical address space with

either 4KB or 4MB page sizes. For modes A and B, the page and directory table entries are each 4 bytes. Mode C supports a 36 bit physical address space for a physical address size of 64GB (physical address extension) with either 4KB or 2MB page sizes. For mode C, the page and directory table entries are each 8 bytes. For each mode, the page and directory tables are equal in size to a page. All modes are for translating 32 bit linear addresses.

[0022] Mode A is illustrated in Fig. 5, where the first 12 bits of a linear address are used as an offset to a physical address within a page frame, the next 10 bits of the linear address are used as an offset into a page table, and the highest 10 bits of the linear address are used as an offset into a page directory. For example, in Fig. 5, PTE 502 in page table 504 pointed to by table field 506 of the linear address provides the address of the desired page frame in physical memory, and when concatenated with offset 508 of the linear address provides the physical address of the desired object. The PDBR register, page directory entries, and page table entries each provide the upper 20 bits of a 32 bit address, so that page directories, page tables, and pages are each forced to be aligned on 4KB boundaries.

[0023] Mode B for 4MB page sizes is illustrated in Fig. 6. For 4KB page sizes, mode B is similar to mode A. The first 22 bits of the linear address provides the offset into a physical 4MB page frame, and the highest 10 bits of the linear address provides the offset into a page table. Note that mode B with 4MB page sizes requires only one level of address translation. A PDE in the page directory of Fig. 6 provides the upper 10 bits of a 32 bit address to force pages to be aligned on 4MB boundaries.

[0024] Mode C for 4KB page sizes is illustrated in Fig. 7. This involves a third level of address translation provided by page directory pointer table (PDPT) 702. Each entry in PDPT 702 is 8 bytes, and there are 4 entries in a PDPT. PDBR 704 provides the upper 27 bits of a 32 bit address pointing to the base of a PDPT so that PDPTs are forced to be aligned on 32 byte boundaries. Each entry in the PDPT, page directory, and page table provides the upper 24 bits of a 36 bit address so that page directories, page tables, and pages are forced to be aligned on 4KB boundaries.

[0025] Mode C for 2MB page sizes is illustrated in Fig. 8. Only two levels of address translation are required, where again a four entry PDPT is used to point to a page directory. Entries in the page directory provide the upper 15 bits of a 36 bit address so that pages are forced to be aligned on 2MB boundaries.

[0026] The page structure described in Figs. 7 and 8 for Mode C allows up to 4GB of the 64GB extended address space to be addressed at one time. To address other 4GB sections of the extended address space, a different entry may be placed in the PDBR register so as to point to a different PDPT, or entries in the PDPT may be changed. Further details of address translation for the IA-32 architecture may be found in the Intel Architecture Developer's Manual for the Pentium® Pro, Vol. 3, available from Intel Corporation. (Pentium® Pro is a registered trademark of Intel Corporation.)

[0027] Increasing the linear address space of a microprocessor provides larger user and system space and reduces the burden associated with linear address exhaustion for a larger physical address space. Increasing the word size of a microprocessor, e.g., from 32 bits to 64 bits, to provide a larger linear address space is a major engineering design task. It may therefore be of economic utility to increase the linear address space of an existing microprocessor design without increasing its word size. Furthermore, it may be advantageous for a microprocessor with increased linear address space to be backward compatible with code designed for the original sized linear address space and supported paging structures.

[0028] EP-A-0530682 discloses ways to obtain control extenders for extending the size of small real and absolute addresses to enable them to locate data or program entities anywhere in a very large memory. The control extenders are concatenated to the high-order end of a conventionally-generated real or absolute address of less than 32 bit size to provide a real/absolute address of greater than 31 bit size.

[0029] US-A-5,566,308 discloses a processor core for providing a linear extension of addressable memory space of a microprocessor with minimal additional hardware and software complexity. A program counter holds an N+x bit instruction address which address provides a pointer to an instruction in the memory to be processed by an execution unit. The reference also discloses an address former which concatenates the portions of the address in two registers to form the data address.

Summary

[0030] According to a first aspect of this invention there is provided a microprocessor as claimed in claim 1 herein.

[0031] According to a second aspect of this invention there is provided a method as claimed in claim 5 herein.

[0032] According to a third aspect of this invention there is provided a computer program as claimed in claim 7 herein.

Brief Description of the Drawings

[0033]

Fig. 1 provides a prior art diagram of a computer system.

Fig. 2 provides a prior art diagram of a microprocessor.

Fig. 3 provides a prior an illustration for translating a logical address to a linear address.

Fig. 4 provides a prior art illustration for translating

a linear address to a physical address.

Fig. 5 provides a prior art illustration for translating a 32 bit linear address to a 32 bit physical address with 4-KByte paging.

Fig. 6 provides a prior art illustration for translating a 32 bit linear address to a 32 bit physical address with 4-MByte paging.

Fig. 7 provides a prior art illustration for translating a 32 bit linear address to a 36 bit physical address with 4-KByte paging.

Fig. 8 provides a prior an illustration for translating a 32 bit linear address to a 36 bit physical address with 2-MByte paging.

Fig. 9 provides an exemplary embodiment for providing an extended linear address.

Fig. 10 provides another exemplary embodiment for providing an extended linear address.

Fig. 10a provides an exemplary implementation of Fig. 10.

Fig. 11 provides an exemplary embodiment for translating a 42 bit linear address to a 36 bit physical address with 4-KByte paging and 4 byte entries.

Fig. 12 provides an exemplary embodiment for translating a 42 bit linear address to a 36 bit physical address with 4-MByte paging and 4 byte entries.

Fig. 13 provides an exemplary embodiment for translating a 42 bit linear address to a 36 bit physical address with 4-KByte paging and 8 byte entries.

Fig. 14 provides an exemplary embodiment for translating a 42 bit linear address to a 36 bit physical address with 2-MByte paging and 8 byte entries.

Detailed Description of Embodiments

[0034] Embodiments of the present invention provide for a backward compatible, extended linear address space by utilizing one or more opcodes to indicate when extended linear addressing is to be utilized. In one embodiment, an opcode indicates whether an LAE (Linear Address Extension) bit in a microprocessor register is to be set. If the LAE bit is set, then AGU **222** translates logical addresses to extended linear addresses.

[0035] Fig. 9 illustrates an arrangement for translating logical addresses to extended linear addresses. Segment register **902** is extended beyond that which is required to select a segment descriptor, as seen in Fig. 9. A portion of segment register **902**, denoted by segment selector **904,** is used to select descriptor table **906** and segment descriptor **908** so as to provide a base address as discussed previously, and an offset value in offset register **910** is added to the base address to provide a lower portion of the extended linear address, indicated by **912.** A portion of segment register **902** not used to select segment descriptor **908,** denoted by segment extension **914,** forms the upper portion of the extended linear address, denoted by **916.** When **914** is added or concatenated with lower portion linear address **912,** the extended linear address is obtained.

[0036] In another arrangement, instructions provide the extended linear address via their source registers, where the extended linear address is obtained by concatenating the values stored in the source registers. For example, a new instructions for loading, storing, adding, and exchanging objects in memory may be introduced in the instruction set. These new instructions are decoded by decoder **206** into one or more microinstructions, where a microinstruction specifies an extended linear address via source registers in its opcode.

[0037] This procedure is illustrated in the flow diagram of Fig. 10. In step **1002,** the values in the source register named by the decoded instruction are concatenated to form the extended linear address, provided the decoded instruction is an instruction that belongs to the set of instructions operating in the extended linear address space. In step **1004**, when a decoded instruction is to operate in the original linear address space, then the offset provided by the instruction is added to the base address provided by the segment descriptor to obtain the linear address.

[0038] Fig. 10a provides an implementation of Fig. 10. Multiplexers **1006, 1008,** and **1010** select their inputs based upon whether there is an extended linear address microinstruction (LAEuop line is asserted). If the LAEuop line is not asserted, multiplexers **1006** and **1008** provide to AGU 222 an instruction queue immediate and segment base address, respectively, so that the linear address may be computed in a conventional manner. If, however, LAEuop is asserted, then multiplexer **1010** provides the concatenation of the contents of registers *R*1 and *R*2 so that an extended linear address is obtained.

[0039] Once an extended linear address is generated, it is translated to a physical linear address. Embodiments of the present invention provide for this address translation by introducing an extra level of translation into the address translation hierarchy, where this extra level of translation is conditionally utilized provided extended linear addressing is indicated.

[0040] Fig. 11 illustrates an embodiment for extended linear address translation with 4KB paging. In the specific example of Fig. 11, an extended linear address is 42 bits, where the highest 10 bits are used as an offset into page directory **1102.** The base address for page directory **1102** is provided by PDBR **1104.** Mux (multiplexer) **1106** is used symbolically in Fig. 11 to indicate that if extended linear addressing is indicated, then the PDE provided by page directory **1102** is used as the base address for the next lower level of address translation, which is page directory **1108.** Extended linear addressing may be indicated, for example, if an LAE bit is set or if not all of the upper 10 bits of the linear address are zero. If extended linear addressing is not supported, then mux **1106** symbolically indicates that PDBR **1104** is used to point to the base address of page directory **1108.** Directory and page table entries are each 4 bytes. PTE **1110** provides the upper 24 bits of a 36 physical

address, which when concatenated with 12 bits from offset **1112** provides a 36 bit address. The physical address space is 64-GBytes.

[0041] An embodiment for extended linear address translation with 4MB paging is shown in Fig. 12. Page directory **1202** provides an extra level of address translation, conditional upon whether extended linear addressing is indicated. Because of the page size, only two levels of page directories are utilized for translating an extended linear address to a physical address. Each page directory entry in Fig. 12 is 4 bytes, and the physical address space is 64-GBytes.

[0042] An embodiment for extended linear address translation with 4KB paging in an extended physical address space of 64-GBytes is shown in Fig. 13. As in mode C in Fig. 7, when extended linear address translation is not indicated, PDBR **1302** is used to point to the base address of PDPT **1304,** which is usually kept in a cache. To support extended linear address translation, PDBR **1302** is used to point to the base address of page directory **1306,** and bit positions 30 through 38 (Addr[38:30]) of the extended linear address provide the offset into page directory **1306.** Note that the first four entries in page directory **1306** are also cached in PDPT **1304.** For Fig. 13, directory and page table entries are each 8 bytes, and the number of entries in each directory and page is $2^9 = 512$ so that each directory and page is 4KB in size. Because only 9 bits are used as an offset into page directory **1306,** bits above position 38 in the linear address are not used in this embodiment. Consequently, if the linear address register is 42 bits, extended address translation is provided for 42 bit linear addresses with the highest three bits equal to zero.

[0043] An embodiment for extended linear address translation with 2MB paging in an extended physical address space of 64-GBytes is shown in Fig. 14, and should be self-explanatory. Entries in the page directories of Fig. 14 are 8 bytes, so that as in Fig. 13 the linear address bits above position 38 are zero.

[0044] Various modifications may be made to the disclosed embodiments without departing from the scope of the invention as claimed below.

Claims

1. A microprocessor comprising:

a decoder to decode instructions so as to provide an offset,
a segment register (902) to store a segment selector (904) and a segment extension (914),
an address generation unit (1006-1010) to generate an extended linear address,
said extended linear address comprising a lower portion and an upper portion, the lower portion being based upon the offset (910) and the segment selector, and the upper portion being based upon the segment extension (914),

wherein during a page walk to translate the extended linear address to a physical address while in the extended linear address mode, the microprocessor utilises a first pointer to reference a first page directory table (1108), said upper portion being used as an offset together with a second pointer (1104) used as a base address to reference a second table (1102) in order to obtain said first pointer, and while not in the extended linear address mode during the page walk the microprocessor utilises said second pointer (1104) to reference said first page directory table wherein said second pointer does not utilise the segment extension.

2. The microprocessor as set forth in claim 1, wherein the address generation unit provides the lower portion as the sum of the offset and a base address, wherein the base address is provided by a segment descriptor from a descriptor table stored in memory and pointed to by the segment selector, wherein the upper portion is equal to the segment extension.

3. The microprocessor as set forth in claim 2, wherein the microprocessor is a 32 bit processor and the extended linear address has more than 32 bits.

4. A computer comprising:

a system bus (14);
memory (112) coupled to the system bus; and
a microprocessor (102) as claimed in any preceding claim coupled to the system bus.

5. A method to perform a page walk, the method comprising:

generating an extended linear address comprising a lower portion based upon an offset and a segment selector, and an upper portion based upon a segment extension;
selecting a first pointer to point to a first page directory table in memory, said upper portion being used as an offset together with a second pointer (1104) used as a base address to reference a second table (1102) in order to obtain said first pointer when in an extended linear address mode; and
selecting said second pointer to point to said first page directory table in memory and not using the segment extension when not in the extended linear address mode.

6. The method as set forth in claim 5, wherein the offset is provided by at least a portion of a segment extension stored in a segment register.

7. A computer program product comprising computer program code means adapted to perform all the steps of claim 5 when run on a computer.

8. A computer program product as claimed in claim 7 when embodied on a computer readable medium.

Patentansprüche

1. Mikroprozessor, der folgendes umfasst:

einen Decodierer zum Decodieren von Befehlen um einen Versatz vorzusehen;
ein Segmentregister (902) zum Speichern eines Segmentselektors (904) und einer Segmenterweiterung (914);
eine Adresserzeugungseinheit (1006-1010) zum Erzeugen einer erweiterten linearen Adresse,

wobei die genannte erweiterte lineare Adresse einen unteren Abschnitt und einen oberen Abschnitt umfasst, wobei der untere Abschnitt auf dem Versatz (910) und dem Segmentselektor basiert, und wobei der obere Abschnitt auf der Segmenterweiterung (914) basiert,

wobei während einem Seitenlauf zur Übersetzung der erweiterten linearen Adresse in eine physikalische Adresse im erweiterten linearen Adressmodus der Mikroprozessor einen ersten Zeiger auf eine ersten Seitenverzeichnistabelle (1108) verwendet, wobei der obere Abschnitt als ein Versatz verwendet wird, in Verbindung mit einem zweiten Zeiger (1104), der als eine Basisadresse für den Verweis auf eine zweite Tabelle (1102) verwendet wird, um den genannten ersten Zeiger zu erhalten, und wobei der Mikroprozessor, wenn er sich während dem Seitenlauf nicht in dem erweiterten linearen Adressmodus befindet, den genannten zweiten Zeiger (1104) für einen Verweis auf das genannte erste Seitenverzeichnis verwendet, wobei der genannte zweite Zeiger die Segmenterweiterung nicht verwendet.

2. Mikroprozessor nach Anspruch 1, wobei die Adresserzeugungseinheit den unteren Abschnitt als die Summe des Versatzes und einer Basisadresse verwendet, wobei die Basisadresse durch einen Segmentdeskriptor aus einer Deskriptortabelle vorgesehen wird, die im Speicher gespeichert ist und auf die der Segmentselektor zeigt, wobei der obere Abschnitt der Segmenterweiterung entspricht.

3. Mikroprozessor nach Anspruch 2, wobei es sich bei dem Mikroprozessor um einen 32-Bit-Prozessor handelt, und wobei die erweiterte lineare Adresse mehr als 32 Bits aufweist.

4. Computer, der folgendes umfasst:

einen Systembus (14);
einen mit dem Systembus gekoppelten Speicher (112); und
einen Mikroprozessor (102) gemäß den vorstehenden Ansprüchen, der mit dem Systembus gekoppelt ist.

5. Verfahren zum Ausführen eines Seitenlaufs, wobei das Verfahren folgendes umfasst:

Erzeugen einer erweiterten linearen Adresse, die einen unteren Abschnitt umfasst, auf der Basis eines Versatzes und eines Segmentselektors, und mit einem oberen Abschnitt auf der Basis einer Segmenterweiterung;
Auswahl eines ersten Zeigers, um auf eine erste Seitenverzeichnistabelle im Speicher zu zeigen, wobei der genannte obere Abschnitt als ein Versatz verwendet wird, in Verbindung mit einem zweiten Zeiger (1104), der als eine Basisadresse zum Verweis auf eine zweite Tabelle (1102) verwendet wird, um den genannten ersten Zeiger zu erhalten, und zwar in einem erweiterten linearen Adressmodus; und
Auswahl des genannten zweiten Zeigers, um auf die genannte erste Seitenverzeichnistabelle im Speicher zu zeigen, und

wobei die Segmenterweiterung nicht verwendet wird, wenn man sich nicht in dem erweiterten linearen Adressmodus befindet.

6. Verfahren nach Anspruch 5, wobei der Versatz durch mindestens einen Abschnitt einer in einem Segmentregister gespeicherten Segmenterweiterung vorgesehen wird.

7. Computerprogrammprodukt, das eine Computerprogrammcodeeinrichtung umfasst, die alle Schritte aus Anspruch 5 ausführen kann, wenn sie auf einem Computer ausgeführt wird.

8. Computerprogrammprodukt nach Anspruch 7, ausgeführt auf einem computerlesbaren Medium.

Revendications

1. Microprocesseur comprenant:

un décodeur pour décoder des instructions afin de fournir un décalage,
un registre de segment (902) pour stocker un sélecteur de segment (904) et une extension de segment (914),
une unité de génération d'adresse (1006-1010)

pour générer une adresse linéaire étendue, ladite adresse linéaire étendue comprenant une partie inférieure et une partie supérieure, la partie inférieure étant basée sur le décalage (910) et le sélecteur de segment, et la partie supérieure étant basée sur l'extension de segment (914),

dans lequel pendant un parcours de page pour traduire l'adresse linéaire étendue en une adresse physique en mode d'adresse linéaire étendue, le microprocesseur utilise un premier pointeur pour référencer une première table de répertoire des pages (1108), ladite partie supérieure étant utilisée comme décalage avec un deuxième pointeur (1104) utilisé comme adresse de base pour référencer une deuxième table (1102) afin d'obtenir ledit premier pointeur, et
lorsqu'il n'est pas en mode d'adresse linéaire étendue pendant le parcours de page le microprocesseur utilise ledit deuxième pointeur (1104) pour référencer ladite première table de répertoire des pages dans lequel ledit deuxième pointeur n'utilise pas l'extension de segment.

2. Microprocesseur selon la revendication 1, dans lequel l'unité de génération d'adresse fournit la partie inférieure comme somme du décalage et d'une adresse de base, dans lequel l'adresse de base est fournie par un descripteur de segment à partir d'une table de descripteurs stockée dans la mémoire et pointée par le sélecteur de segment, dans lequel la partie supérieure est égale à l'extension de segment.

3. Microprocesseur selon la revendication 2, dans lequel le microprocesseur est un processeur 32 bits et l'adresse linéaire étendue a plus de 32 bits.

4. Ordinateur comprenant:

un bus système (14);
une mémoire (112) couplée au bus système; et un microprocesseur (102) selon l'une quelconque des revendications précédentes couplé au bus système.

5. Procédé pour réaliser un parcours de page, le procédé comprenant :

la génération d'une adresse linéaire étendue comprenant une partie inférieure basée sur un décalage et un sélecteur de segment, et une partie supérieure basée sur une extension de segment ;
la sélection d'un premier pointeur pour pointer une première table de répertoire des pages dans la mémoire, ladite partie supérieure étant utilisée comme décalage avec un deuxième pointeur (1104) utilisé comme adresse de base pour référencer une deuxième table (1102) afin d'obtenir ledit premier pointeur en mode d'adresse linéaire étendue ; et
la sélection dudit deuxième pointeur pour pointer ladite première table de répertoire des pages et ne pas utiliser l'extension de segment en dehors du mode d'adresse linéaire étendue.

6. Procédé selon la revendication 5, dans lequel le décalage est fourni par au moins une partie d'une extension de segment stockée dans un registre de segment.

7. Produit de programme informatique comprenant des moyens formant codes de programme informatique adaptés pour réaliser toutes les étapes de la revendication 5 lorsqu'ils sont exécutés sur un ordinateur.

8. Produit de programme informatique selon la revendication 7 lorsqu'il est réalisé sur un support lisible par ordinateur.

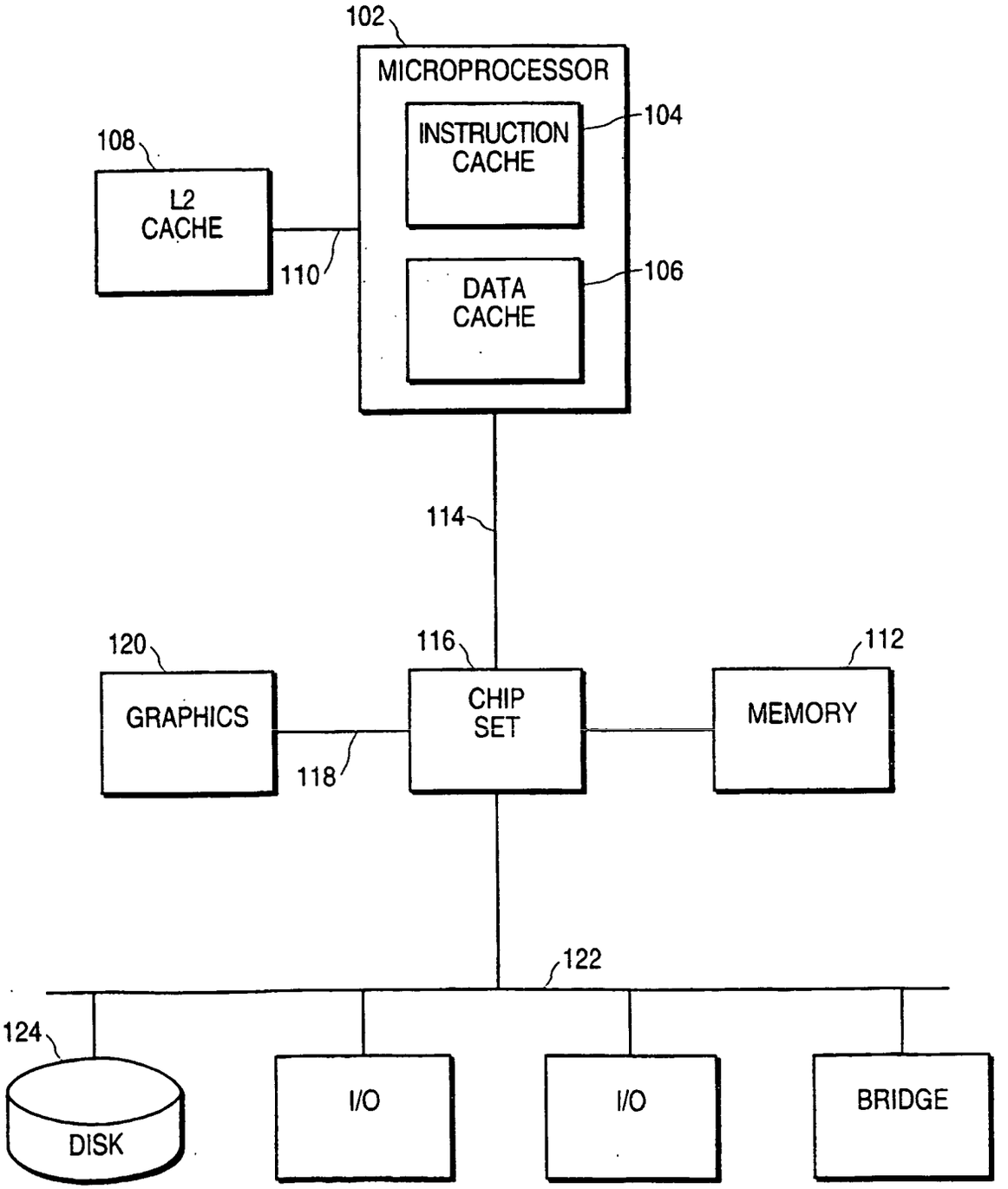

FIG.1 (PRIOR ART)

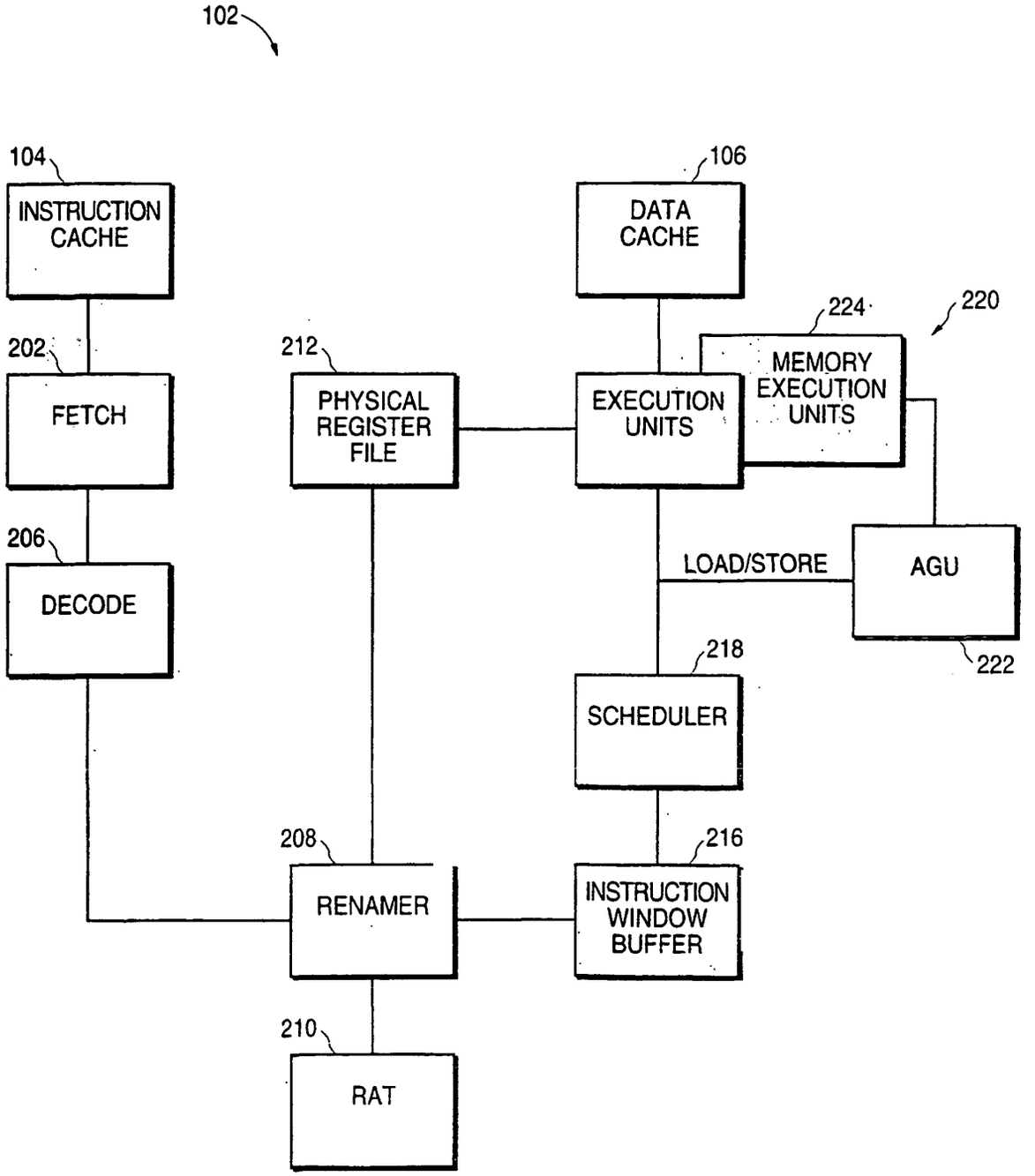

FIG.2 (PRIOR ART)

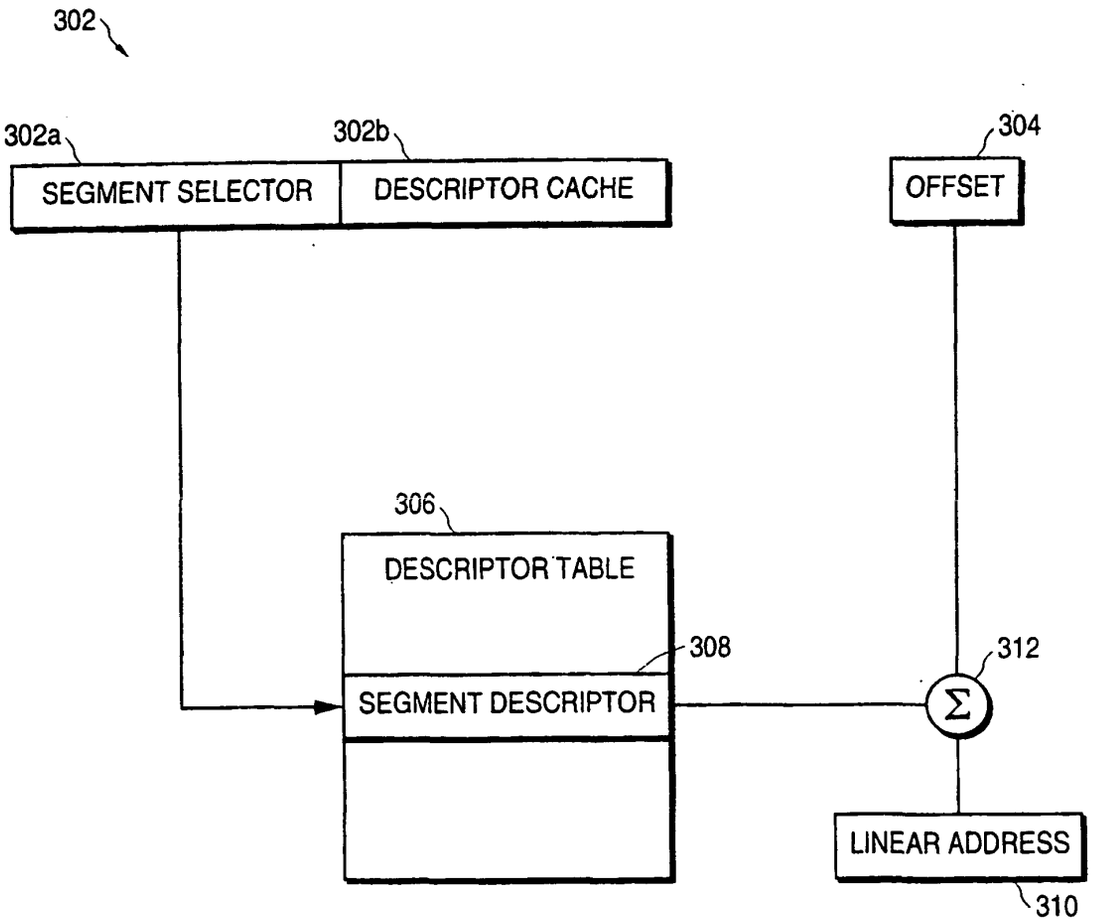

FIG.3 (PRIOR ART)

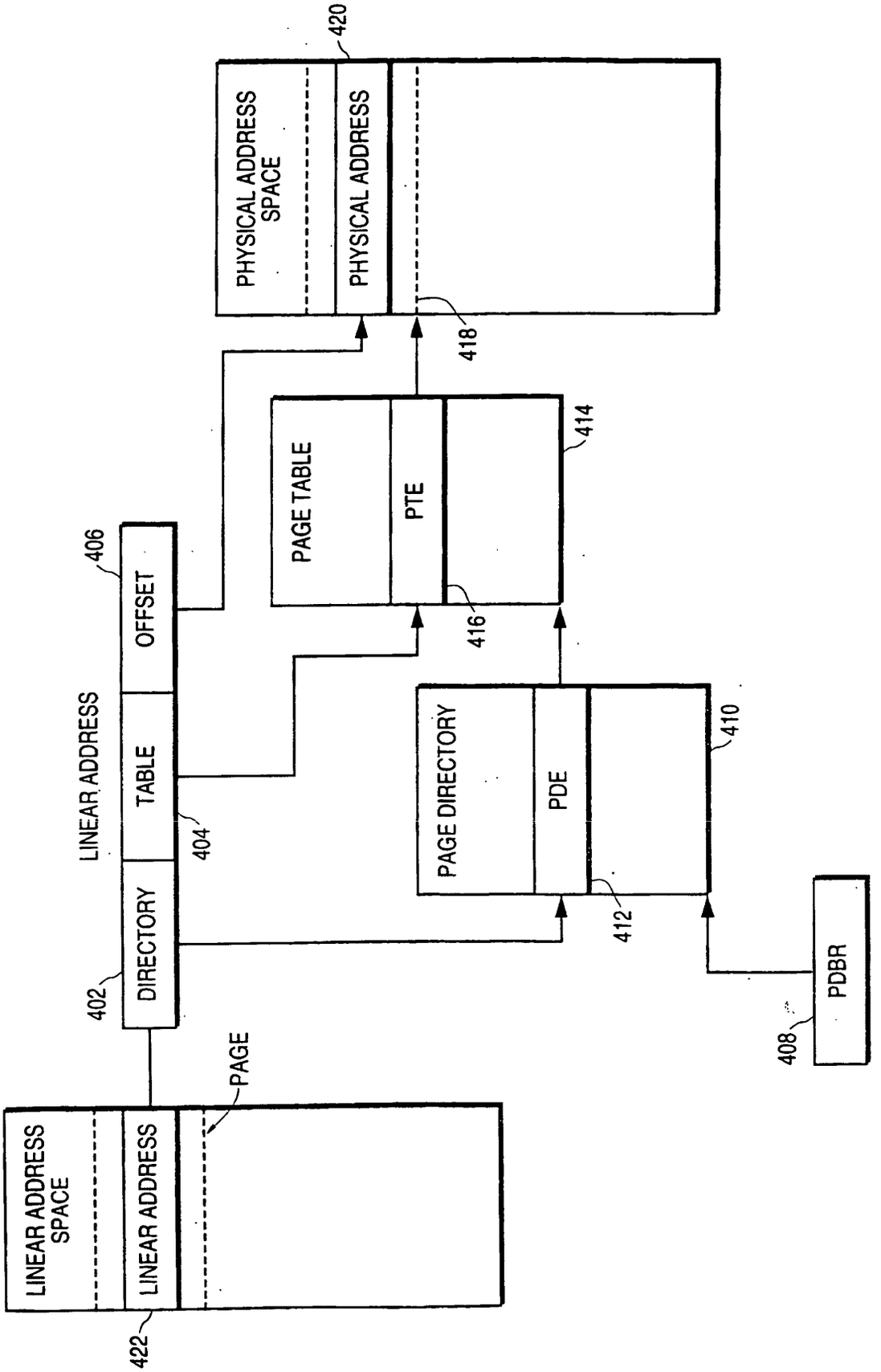

FIG.4 (PRIOR ART)

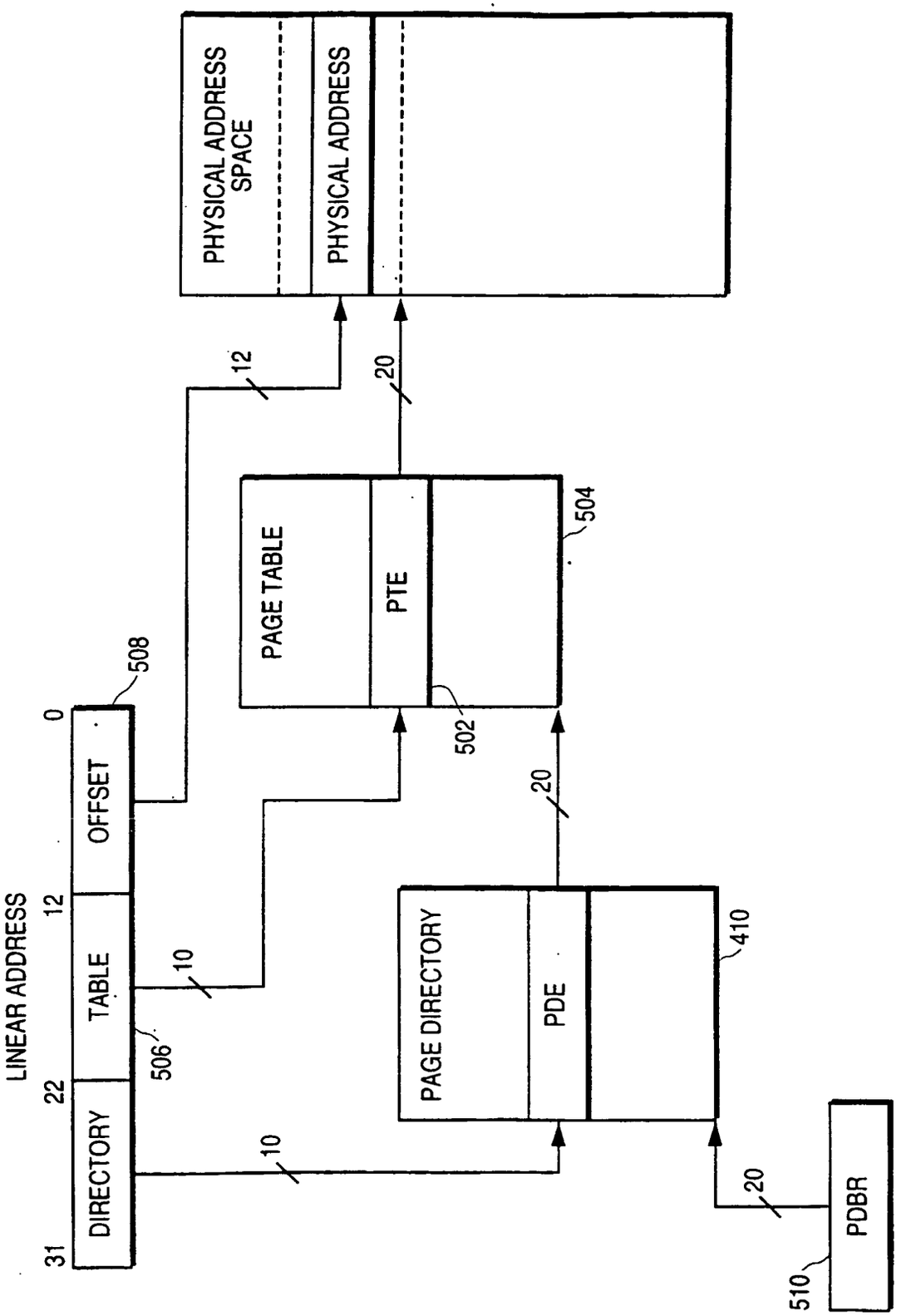

FIG.5 (PRIOR ART)

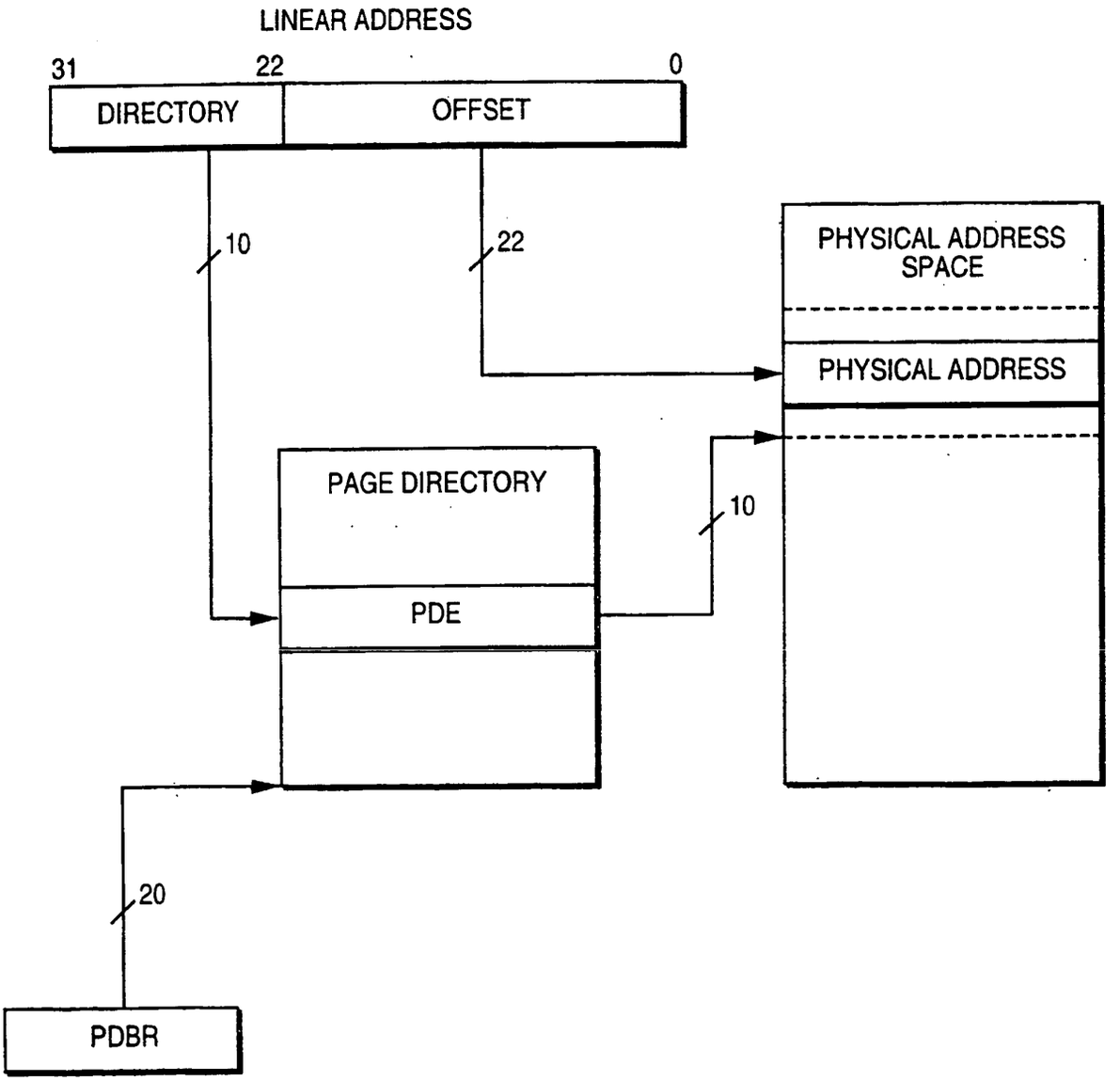

FIG.6 (PRIOR ART)

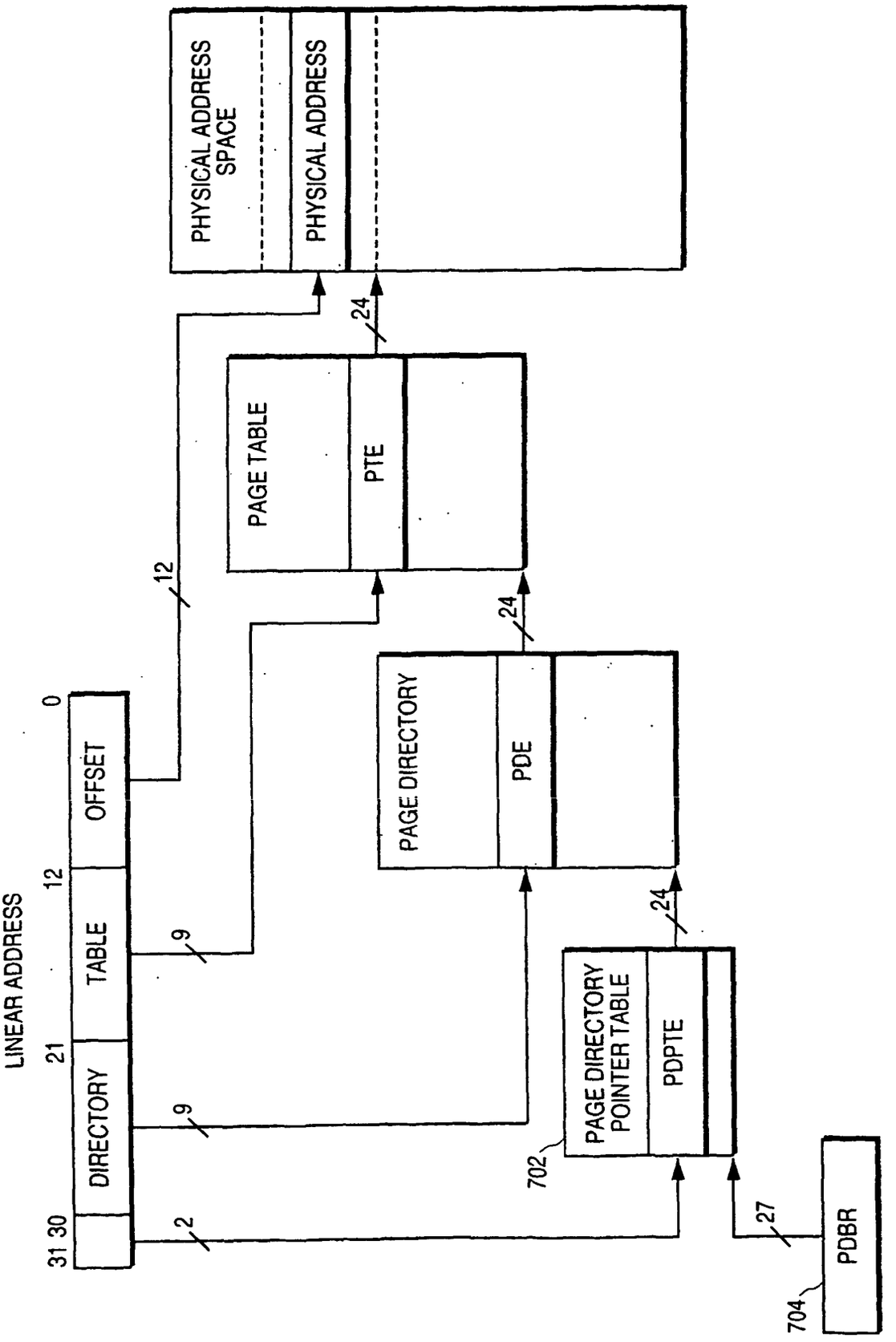

FIG.7 (PRIOR ART)

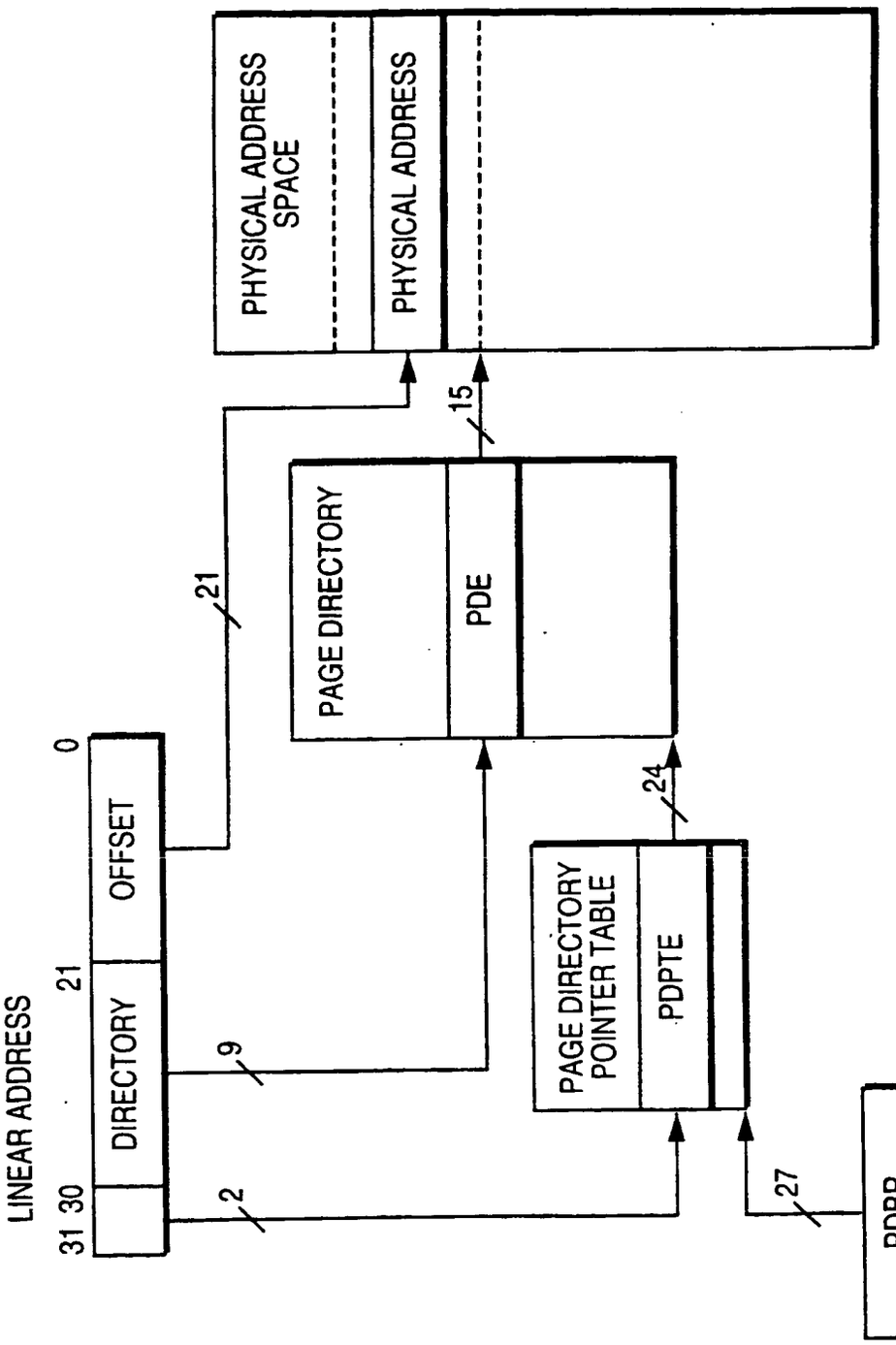

FIG.8 (PRIOR ART)

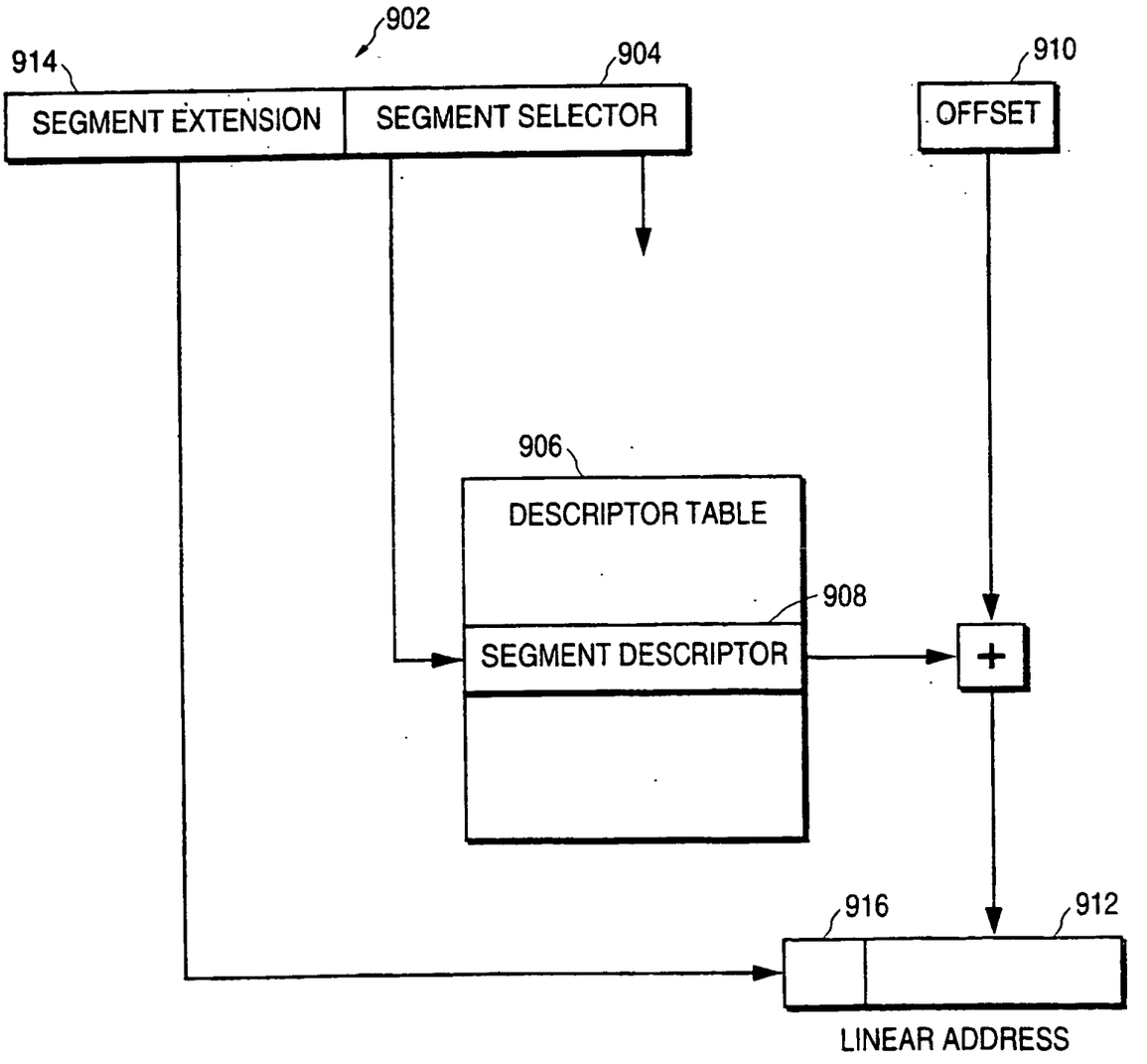
902
914
904
910
SEGMENT EXTENSION
SEGMENT SELECTOR
OFFSET
906
DESCRIPTOR TABLE
908
SEGMENT DESCRIPTOR
+
916
912
LINEAR ADDRESS

FIG.9

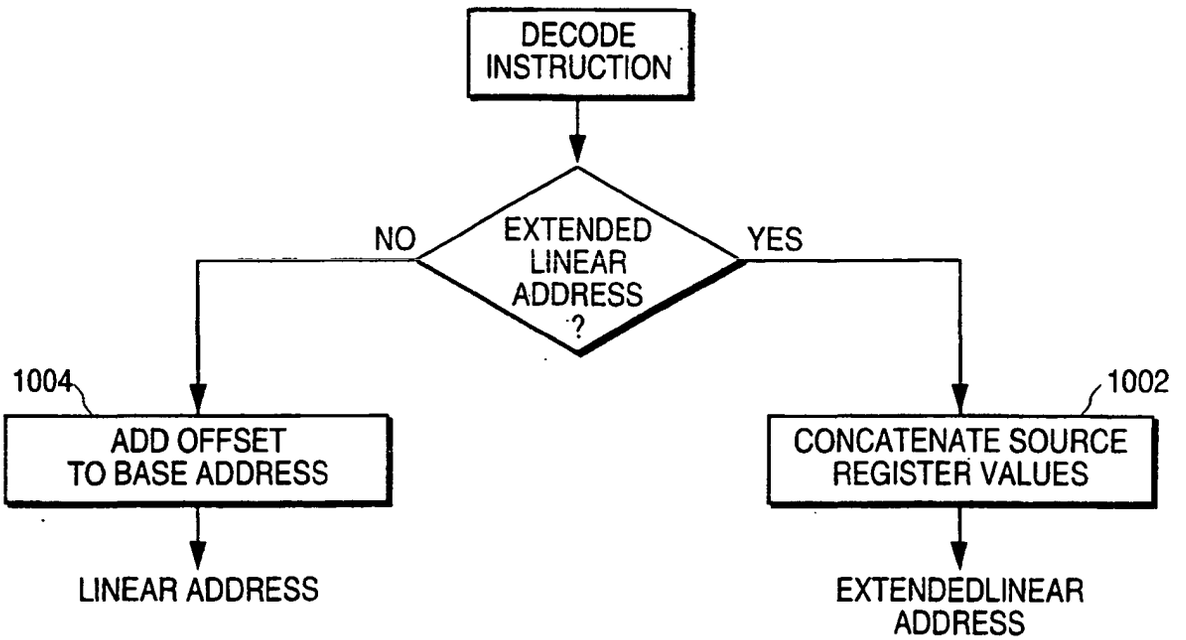
DECODE INSTRUCTION
EXTENDED LINEAR ADDRESS ?
NO
YES
1004
ADD OFFSET TO BASE ADDRESS
LINEAR ADDRESS
1002
CONCATENATE SOURCE REGISTER VALUES
EXTENDEDLINEAR ADDRESS

FIG.10

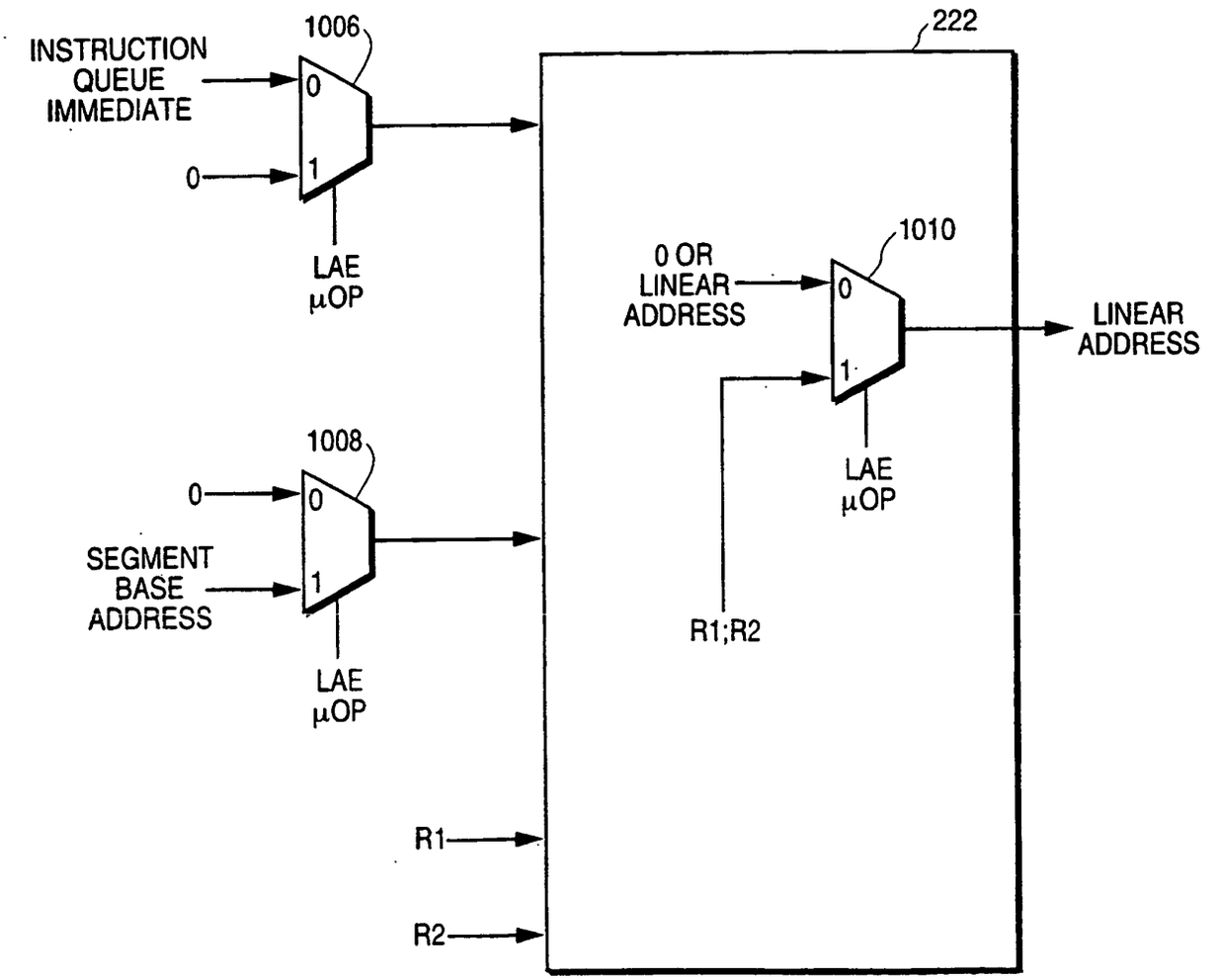
INSTRUCTION QUEUE IMMEDIATE
1006
0
1
0
LAE μOP
222
0 OR LINEAR ADDRESS
1010
0
1
LINEAR ADDRESS
LAE μOP
R1;R2
1008
0
0
1
SEGMENT BASE ADDRESS
LAE μOP
R1
R2

FIG.10A

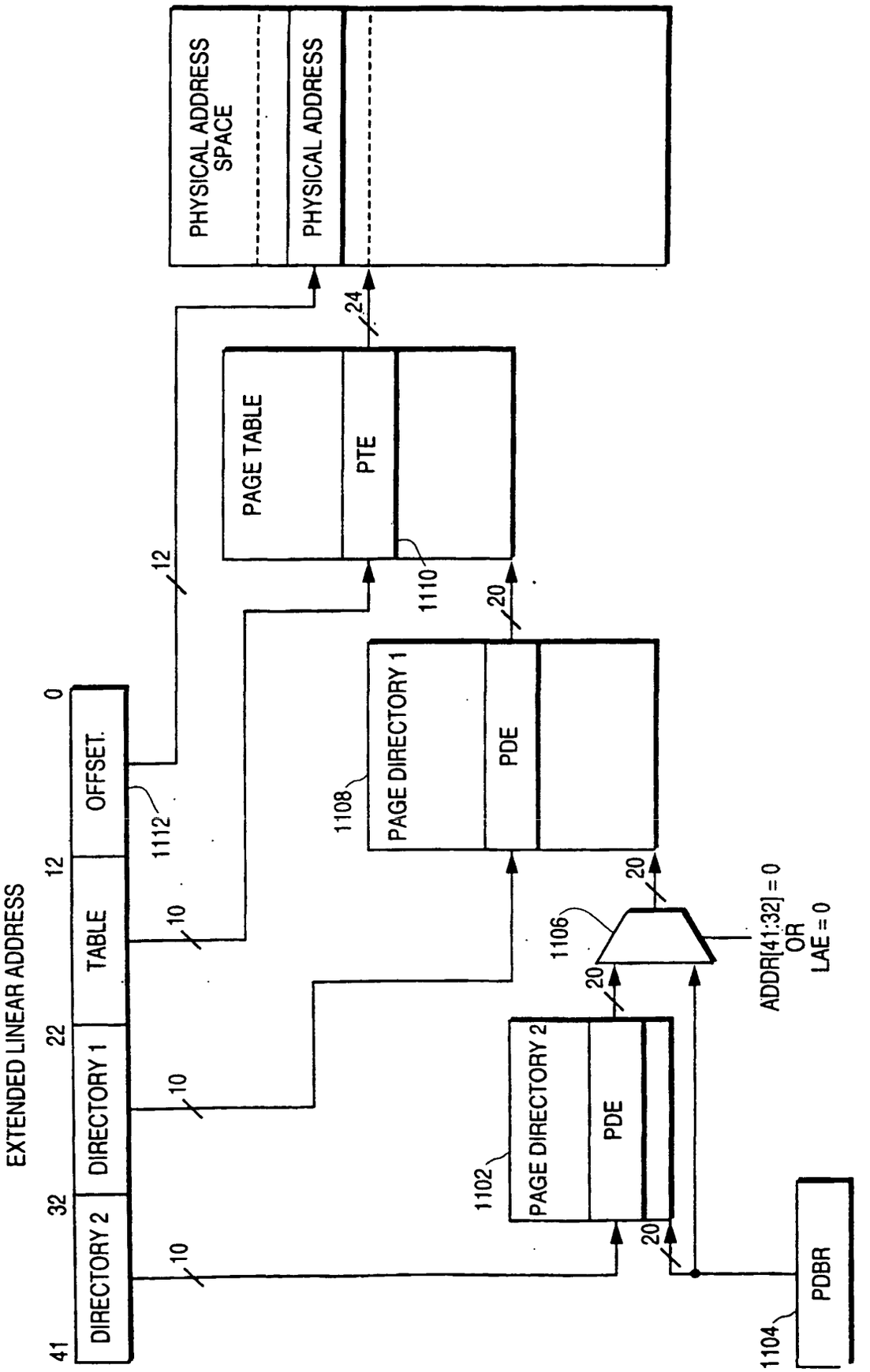
EXTENDED LINEAR ADDRESS
41
32
22
12
0
DIRECTORY 2
DIRECTORY 1
TABLE
OFFSET.
10
10
10
12
1112
1102
PAGE DIRECTORY 2
PDE
1104
PDBR
20
20
1106
20
ADDR[41:32] = 0
OR
LAE = 0
1108
PAGE DIRECTORY 1
PDE
20
1110
PAGE TABLE
PTE
24
PHYSICAL ADDRESS SPACE
PHYSICAL ADDRESS

FIG. 11

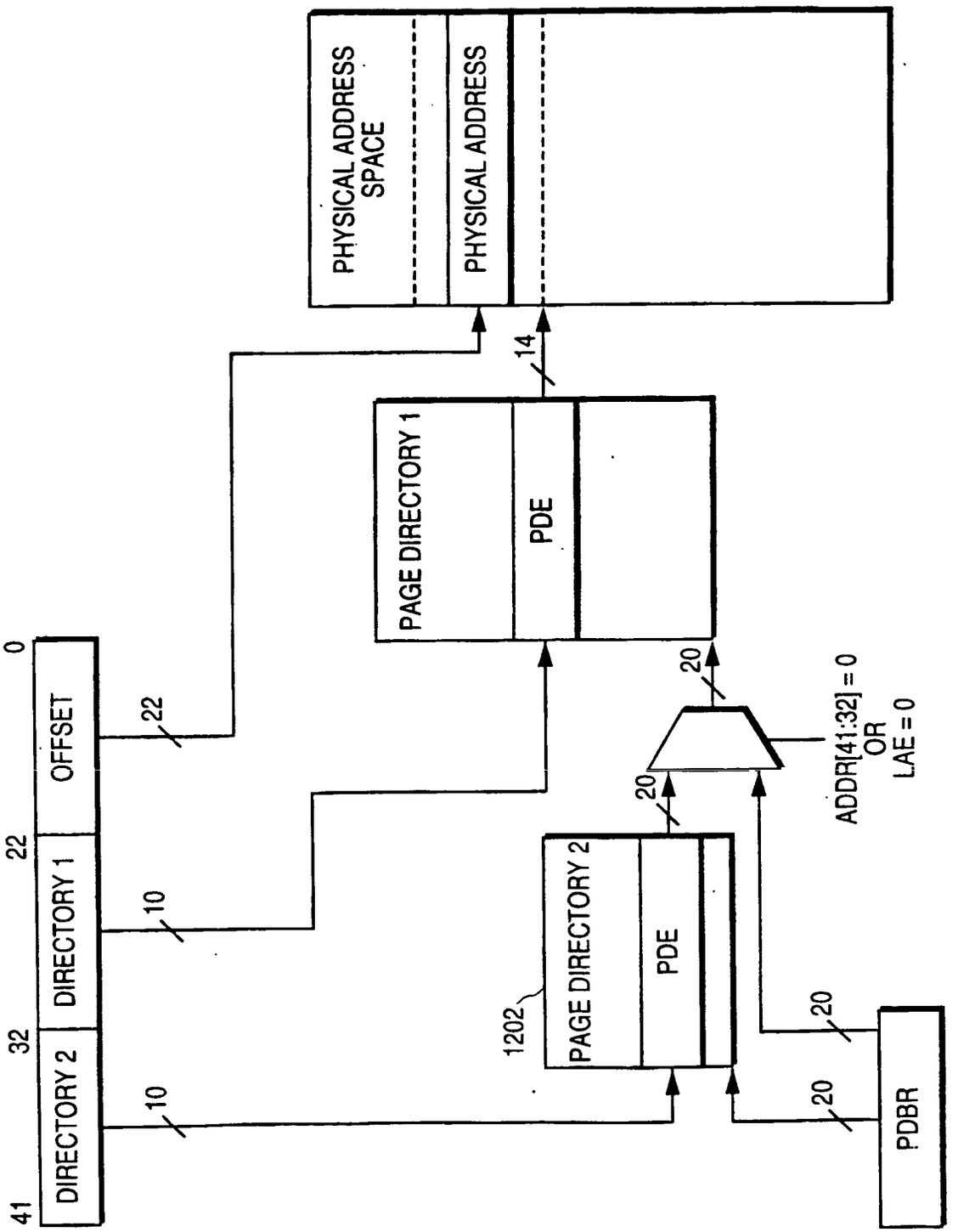
PHYSICAL ADDRESS SPACE
PHYSICAL ADDRESS
14
PAGE DIRECTORY 1
PDE
20
ADDR[41:32] = 0
OR
LAE = 0
20
1202
PAGE DIRECTORY 2
PDE
20
20
PDBR
0
OFFSET
22
22
DIRECTORY 1
10
32
DIRECTORY 2
10
41

FIG.12

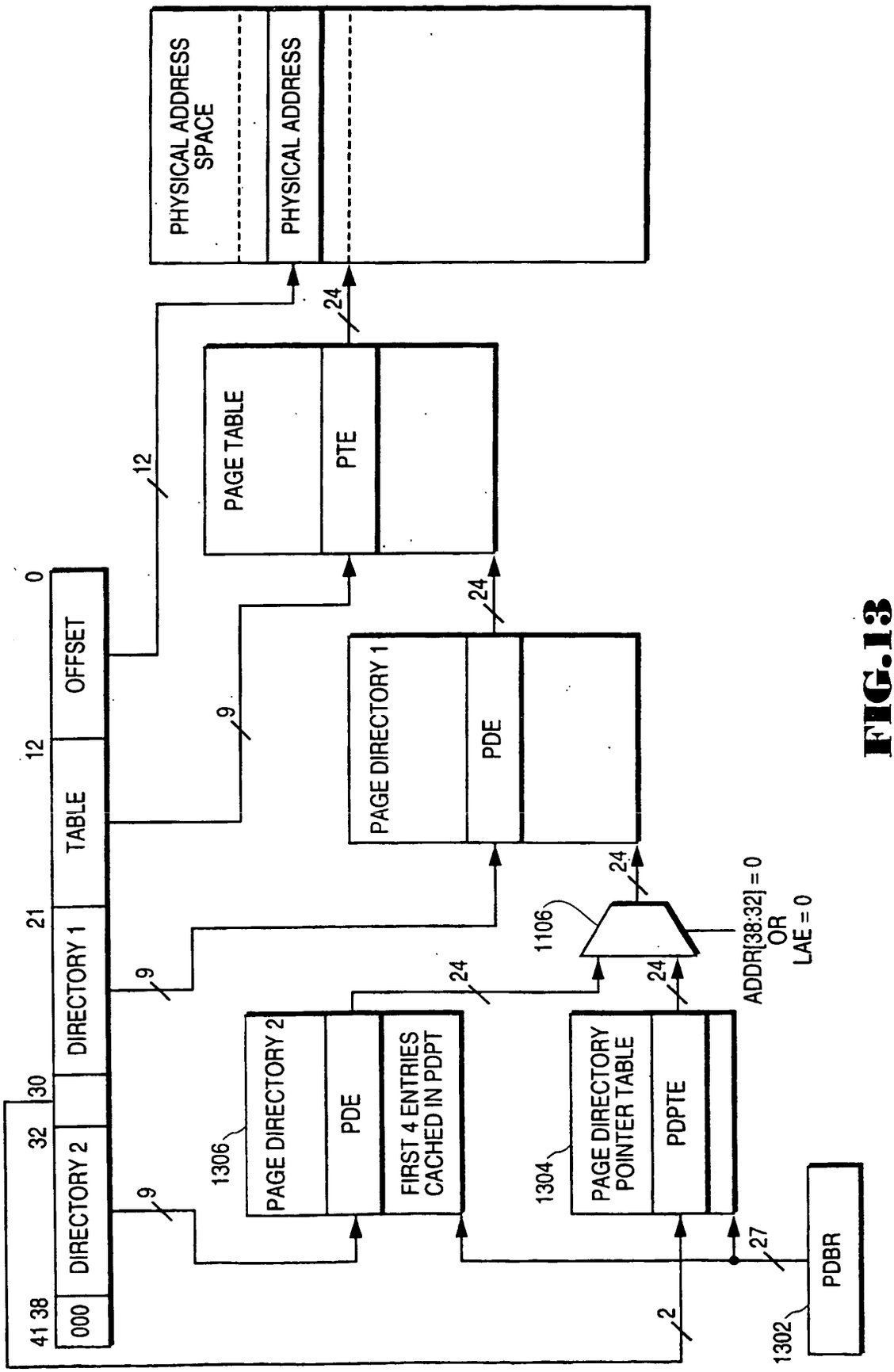
PHYSICAL ADDRESS SPACE
PHYSICAL ADDRESS
PAGE TABLE
PTE
PAGE DIRECTORY 1
PDE
PAGE DIRECTORY 2
PDE
FIRST 4 ENTRIES CACHED IN PDPT
PAGE DIRECTORY POINTER TABLE
PDPTE
PDBR
1106
1302
1304
1306
ADDR[38:32] = 0 OR LAE = 0
41 38
000
DIRECTORY 2
32
30
DIRECTORY 1
21
TABLE
12
OFFSET
0
12
9
24
2
27

FIG.13

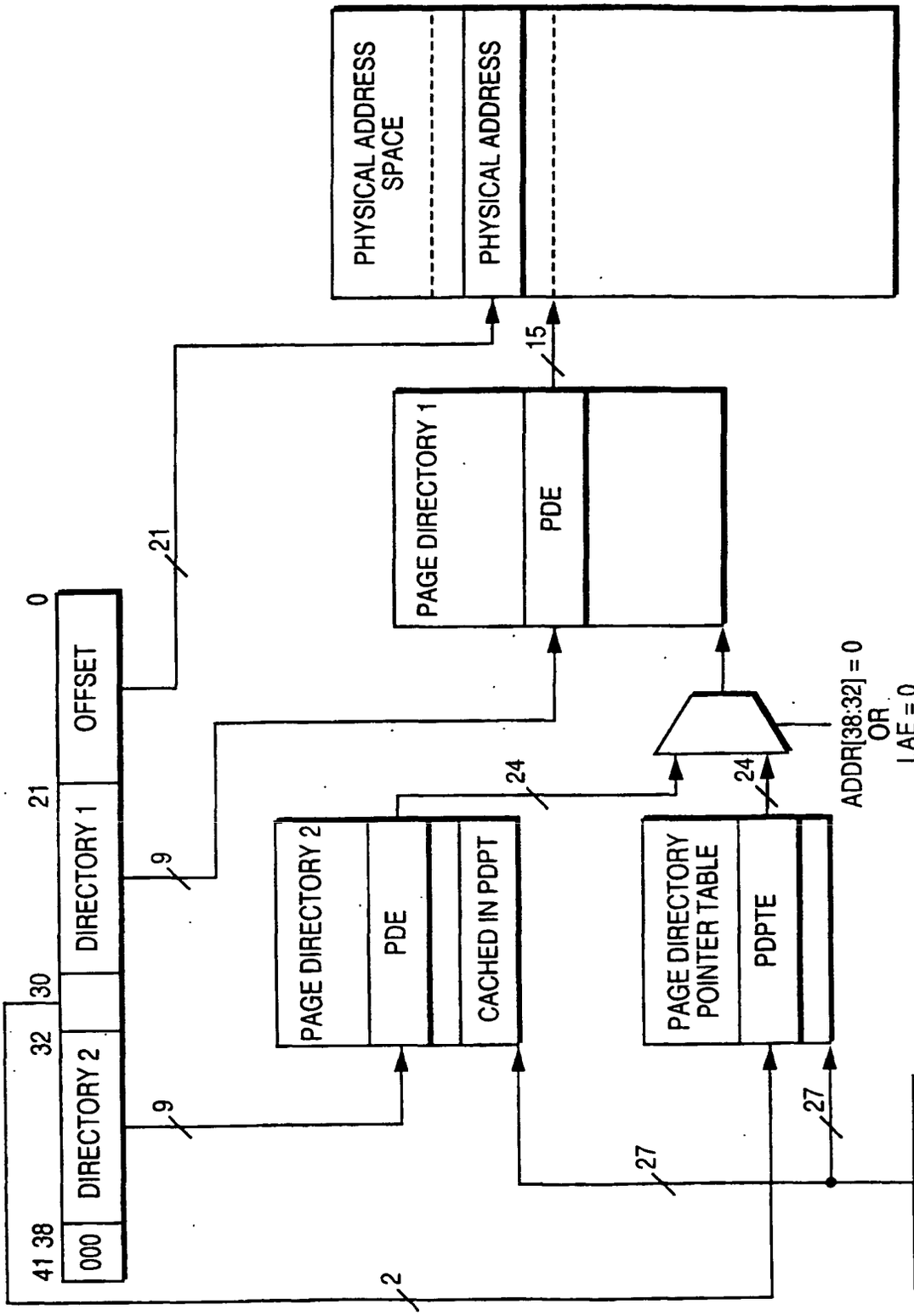
PHYSICAL ADDRESS SPACE
PHYSICAL ADDRESS
PAGE DIRECTORY 1
PDE
15
21
0
OFFSET
21
DIRECTORY 1
9
30
32
DIRECTORY 2
9
41 38
000
2
PAGE DIRECTORY 2
PDE
CACHED IN PDPT
24
PAGE DIRECTORY POINTER TABLE
PDPTE
24
ADDR[38:32] = 0
OR
LAE = 0
27
27
PDBR

FIG.14