# United States Statutory Invention Registration 

[11]

Reg. Number:
H472
[43] Published:
May 3, 1988
[54] METHOD AND APPARATUS FOR PROCESSING BINARY-CODED/PACKED DECTMAL DATA
[76] Inventor: Ralph W. Peterson, 6S556 Sussex
Rd., Naperville, III. 60540
[2i] Appl. No.: 2,428
[22] Filed: Jan. 12, 1987
[si] Int. Cl. ${ }^{4}$
G06F 1/00; GOSF 7/50
[s2] U.S. C.
364/200; 364/771

## References Cited

U.S. PATENT DOCUMENTS
3.937,941 2/1976 Zemel et al. ........................ 235/159

## FOREIGN PATENT DOCUMENTS

3303316 B/1983 Fed. Rep. of Germany 2115962 9/1983 United Kingdom

## OTHER PUBLICATIONS

R. Zaks, Programming the 280 (Sybex, 1982) pp. 107-113, 236-237.
J. A. Otto, "Predicting Potential COBOL Performance on Low Level Mechine Architectures," SIGPLAN Notices, vol. 20, No. 10.
R. K. Richards, Arithmetic Operations in Digital Computers (D. Van Nostrand Co. 1935), pp. 209-285.
G. Chroust, "Method of Adding Decimal Numbers by Mcans of Binary Arithmetic", IBM Technical Disclosure Bulletin, vol. 23, No. 10 (3-81).
D. R. Hicks et al., "Multidigit Decimal Addition and Subtraction", IBM Trehnical Disclosure Bulletim, nol 19. Na 11 (4-77).

## Assistant Examiner-Linda J. Wallace

## [57]

## ABSTRACT

A system (FIG. 1; FIGS. 18-19) performs addition or subtraction of packed, or binary-coded-decimal (BCD), values. Each BCD digit is stored in a nibble (500) of a register ( 400,401 ). The least-significant bits (LSBs) (502) of corresponding nibbles of the registers are cxclu-sive-ORed and results are stored in a third register (402) (FIG. 4 or 9). For addition, the registers' binary valucs are summed, six is added to each nibble of the sum using binary addition, and results are stored in one register (401) (FIG. 5). For subtraction, the registers' binary values are subtracted, and the results are stored in one register (401) (FIG. 10). The LSB of each nibble of the one register is compared with the corresponding exclu-sive-OR value from the third register (FIG. 6 or 11). For every comparison that indicates equality for addition, and that indicates inequality for subtraction, six is subtracted using binary subtraction from the value of the one register's nibble that precedes the compared values' corresponding nibble in the registers (FIGS. 6-7 or 11-12).

## 52 Claims, 15 Drawing Sheets

A statutory lavention registration is not a patent. It has the defensive attributes of a patent but does not have the enforceable attributes of a patent. No article or advertisement or the lille may use the term patent, or any term suggestive of a patent, when referring to a statutory invention regiatration. For more specific information on the rights aseocinted with a statutory invention registration mee 35 U.S.C. 157.


FIG. 1
U.S. Patent May 3, $1988 \quad$ Sheet 2 of $15 \quad$ H472


$$
\text { FIG. } 3
$$


U.S. Patent May 3, 1988

REG A

REG B

REG C


FIG. 4

REO A

$+$

| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |



FIG. 5


FIG. 6

REO 1


| $0 / 6$ | $0 / 6$ | $0 / 6$ | $0 / 6$ |
| :---: | :---: | :---: | :---: |

REO B


FIG. 7

FIG. 8


## U.S. Patent May 3, 1988

REG A

REG 日

REG $C$


FIG. 9

REO A

$-400$ —


FIG. 10


FIG. 11

REO B


REO B


FIG. 12

$$
\text { FIG. } 13
$$





FIG. 16


FIG. 17


FIG. 20

| $F 10.18$ | $F 10.18$ |
| :--- | :--- |


FIG. 18

FIG. 19

## METHOD AND APPARATUS FOR PROCESSING BINARY-CODED/PACKED DECIMAL DATA

## TECHNICAL FIELD

The invention is directed to the field of general purpose digital computers in general, and in particular concerns the efficient processing in such computers of data represented in binary-coded, or packed, decimal form and other non-base-2 forms.

## BACKGROUND OF THE INVENTION

The COBOL programming language defines a binary coded decimal data type, which encodes each decimal digit in binary form as a separate character. The traditional decimal data type has been the "unpacked", or ASCII, form, wherein commonly eight bits are used to represent each decimal digit: four bits to hold the value of the digit, and the other four bita to hold a code identifying the character as representing a digit.

This form of the decimal data type is inefficient. It consumea eight bita to hold only a four-bit value, and hence is wasteful of memory. The large number of bits that must be moved each time a digit is moved inside a processor also adversely impactis system performance. To avoid such tnefficiency, a "packed" form of the decimal data type had subsequently been developed, which discarde the code-bolding bita and merely retains the four value-holding bits of the "unpacked" form. The "packed" form is also known as the "binary-coded decimal", or BCD, form; "packed" is otten used wrefer to signed BCD, whereas BCD is typically considered to be unsigned, Le., a magnitude.

Because mout digital computers are binary computers that perform btary, as opponed to decimal, operations, it has conventionally been necessary to convert decimal data lnto binary data before procesaing, such as artihmetic operations, could be performed on that datin on mont computer. But the conversion is time-consuming and impects adversely system performance. A scheme was developed that allowed erithmode operations to be performed on the unpacked decimal digits whthort converion. But this tebeme required the presence of the code-bolding bles for bti-manipulation and temporary storage parpones. Hence, it in not usable with the packed dectmal form.

The prior art has attempted to develop a scbeme that would allow proceming of pecked decimal digits df rectly, without corversion to dither the unpecked dectmal or binary form. The resulta have been disappotating. The scheme requires that epecial hardware be prese ent in a computer for fis support. Hence, the scheme is not arited for uee on general purpione comproters that do not provide that special hardware support. Further more, the achean is cumbersome and meficient, and thereby segpates fo large mearore the benefits of tmproved syitem performance that were aought to be schieved thereby.

Computers have aloo been developed that provide full hardware support to their arithmetio and logic units for both dectmal and blaary procesalag. However, such computers effectively provide eeparate arthmetic and logic unith for tha decimal and Binary data typen. Such duplication as very expenalve. And attempts at avoidiag complete duplication of artehmetic and logic unit hardware by sharing hardware components between the
units introduce undesirable performance-robbing delays into the operation of the arithmetic and logic units.

In summary, then, problems in the art are the lack of a scheme for processing packed decimal data directly, without need for conversion to a different data type, on computers providing no special hardware support therefor, and the lack of a hardware-supported scheme for directly processing packed decimal data that is efficient in terms of both cost and performance.

## SUMMARY OF THE INVENTION

This invention is directed to solving these and other problems of the art. According to the invention, combinatorial processing of binary-coded-decimal values, which take the form of two signals each having sequen-tially-ordered data fields each of which includes a plurality of bits-including a least-significant bit-that define a BCD value, is as follows. If the desired processing involves adding the BCD values defined by the two signale, the binary values of the data fields of the same sequential order are summed with each other and with a value that comprises a plurality of sixes, one six for data fields of the same sequential order. If the desired processing involves subtracting the BCD values, the binary values of the data fields of the same sequential order are subtracted one from the other. Both cases yield a resultant signal that defines a value which represents the resuit of the performed operation (addition or subtraction). The resultant signal has a plurality of data fields ordered correspondingly to the data fields of the initial two signals, and each field includes a plurality of rio, including a least-significant blt. Values of least-sig. aificant bits of data fields of the same sequential or-der-illustratively of all data fields but those of the least algrafficance in the sequential order-of the initial two signals and of the resultant signal are examined. For every examination of least-significant bit values that include an odd number of zeros in the case of the addition operation, and for every examination of least-sigalficant bit values that include an odd number of ones in the case of the subtraction operation, a six is subtracted from the value of the resultant signal's data field that precedes the compared values in the sequential order of data felde. The values of the data fields of the resultant dignal now represent the BCD values that are the com-blation-sum or difference-of the BCD values of the initial two dignala.

Furthermore, determination is llustratively made whether the exmming or difference operation that produced the resultant dignal yielded a carry. If the desired procestas involves eddiag the BCD values of the recelved two dignals, a six is subtracted from the value of the mort dgrificant data field of the resultant signal if a carry in determined not to have been yielded. If the desired procesaing tavolven subtracting the BCD valven, adx is subtricted from the value of the most significant data field of the resultant signal if a carry is deter. mined to have been ylelded.

The decimal viluea are processed without conversion thereof to a differeat data type, yet all of the operations performed on the decimal values are operations whose performance lles within the capability of a binary gen. eral purpose computer. Hence, the processing may be performed on computers having no hardware suppor for decimal data type proceasiag. Consequently, a sys. tem for combinatorily procesaing decimal values mas be based on such computers.

Also, providing that its registers and arithmetic and logic unit have a width, in terms of bits, greater than the width of a coded decimal digit, the computer is able to process a plurality of decimal digits in parallel. Processing of decimal digits may consequently be accomplished much faster than if each decimal digit had to be processed sequentially.
Furthermore, hardware support may be provided in a computer for performing the processing-particularly the comparison, carry-checking, and sixes-subtraction operations-which enables the processing to be performed very quickly and efficiently. The hardware support is structurally simple, and easy to incorporate into existing computer designs. It thus avoids the complexity of structure and function, and hence the cumbersomeness and cost, of preceding attempts at hardwareassisted BCD procesaing.

In an illustrative embodiment, for purposes of ease of implementation, comparing of least-ignificent bits of the signals to repleced by exclusive-ORing of binary values of the least-aignificant bita of data ficids of the same sequential order, to obtain exclusive-OR values each correaponding with data fields of different sequential order, and the values of least-aigolficant bits of data fields of the resultant sigmal are compared each with the corresponding excluaive-OR value. Then, for every comparison indicating equality in the case of the addition operadora, and for every comparison tndicating inequality in the case of the subtraction operation, a atx is subtracted from the value of the resultant signal's data field that precedea the compared values in the sequeptial order of daca felda.

Method and apparatus-whether of resource allocstion, code compilation, or computer operation-according to the linvention at characterized above need not be limited to procesaing of BCD values, but may be applied to procematas of values having a bese other than ten. Assuming that binary-coded bese-(2n) values take the form of dignals each having a plurality of sequeatia-ly-ordered data ficlda each one of which tacludes im blts, where $m$ and $a$ are podtive integers such that $2 m>2 \mathrm{a}_{5}$ the proceming es described above holda true with the exception that the dres recited in the description are replaced with values equal to $\left(2^{m}-2 n\right)$. The method and apparatus of the tavention thus have general applicability to the procesing of nop-binary valuet, yet rotain the full spectrum of thetr advantagen with respect to son-decimal oven-sedix numbers.
These and other advantagen and features of the present lavention will become epparent from the following description of en mustrative embodimeat of the invertion takea together with the drawing.

## BRIEF DESCRIPIION OF THE DRAWINO

FIG. 1 la a block diagram of en illustrative computer system tacludtas an embodiment of the tavention;
FIG. 2 is a tiow diagran of a repource allocadon function performed by the nyteen of FIG. 1;

FIG. 3 ta a flow diagram of the logical functions of a 60 BCD valua eummation process performed by the syttem of FIO. 1 Mlutratively as part of performance of the function of FIO. 2;

FIOS. 4-7 are block diagrams of register, manipuletions oceurriag th the system of FIO. 1 an a conse- 6 quence of the performaces of the process of FIG. 3:

FIO. It a flow diagram of the logical function of a BCD difference-producing process performed by the
system of FIG. 1 illustratively as part of performance o the function of FIG. 2;
FIGS. 9-12 are block diagrams of register manipula tions occurring in the system of FIG. 1 as a conse quence of the performance of the process of FIG. 8;

FIGS. 13-17 are flow diagrams of code for the pro cesses of FIGS. 3 and 8 compiled by the compiler of the system of FIG. 1;
FIGS. 18-19 are a block diagram of structural addi tions made to the CPU of the system of FIG. 1 to pro vide hardware support therein for the processes o FIOS. 3 and 8 ; and
FIG. 20 is a composite showing the arrangement o FIGS. 18-19 to form a single diagram.

## DETAILED DESCRIPTION

FIG. 1 ta a block diagram of a general purpose com puter programmed to perform business functions, in cluding resource allocation functions, such as payrol functiona. Such computers and programs therefor are well known ta the art. The computer comprises a pro cessor 12 that performs data processing functions. Pro cessor 12 includes a central processing unit (CPU) 1 ! which performs operations on data 19 stored in a mem ory 16 eccording to program instructions 17 also storec In memory 16. A terminal 11 connected to processor 1: is used to provide data and control lnput to processo: 12. A printer 13 connected to processor 12 is used tc output remults of deta processing functions from proces eor 12.

An mustrative example of a conventional payrol function performed by the computer of FIG. 1 is flow charted in FIO. 2 and is described next in conjunction with FIG. 1. A termanal 11 operator (not shown) direct procesor 12 to execute the payroll program, and thei takes employes time cards 10 and enters information therefrom into the computer via terminal 11. Alterna tively, information from the time cards may be read ints the computer by means of a card reader (not shown) Entered faformacion tacludes the employec's identifica tion-his or ber name, for example-the number o regular bours worked, and the number of overtim. hours worked.

The payroil program begias to execute on processo 12. at step 200. Processor 12 receives the employe Ideatification, the number of regular hours worked, ani the number of overtime hours worked that have bee entered oa terminal 11, at stepe 201-203, respectivels Proceseor 12 then identified employee's file in memor 16 to obtain fuformation on that employee's standar. homily rate of pay, the overtime rate of pay, and payro deductions belag made for that employee, at step 20 C Proceseor 12 mess CPU 15 to muldiply the regular hour worked by the employee by the standard hourly rate I obtuin the employee's standard pay, at step 205. Proce: sor 12 dimilarly multiplice the overtime hours worke by the employee by the overtime hourly rate to obtai the employee's overtime pry, at step 205. Processor 1 nea CPU 15 to add the ithadard pay to the overtim pay to obtain the employee's grons pay, st step 20 Processor 12 uses CPU 15 to subtract the employec deductom from the gross pay to obtain the employee net pay, at mep 203. Processor 12 then accesses in men ory 16 the employer's payroll account balance, an subtracts therefrom the employec's net pay, at step 20 Finally, processor 12 causet priater 13 to print a pa: check 14 ta the employeo's name in the amount of tt net pay, at step 210.

The net result of the processing that processor 12 has performed has been to reallocate the amount of net pay from the employer's account to the employee. Processor 12 now checks whether input on other employees is being received, at step 211. If so, processor 12 returns to step 201 to repeat the processing for another employee. If not, processor 12 stops executing the program, at step 212.

Business programs, like the payroll program just described, are very often written in the COBOL programming language. COBOL is a "high level" language: programs written in COBOL cannot be executed by a processor directly, but must first be converted into machine language understandable to the processor. This conversion, called compiling, is performed by compilers, in a conventional and well-known manner. Illustri. tively, a compiler 18 may be a program stored in memory 16 and executing on CPU 15 of processor 12.

The COBOL programming language supports the packed dectmal data type. Data nued by COBOL pro- 20 grams-such as hours worked, rate of pay, payroll balance, deductions, and net pay in the example of FIG. 2-may be expressed in processor 16 in packed decimal form. But conventional processors perform binary arithmetic. Hence, when compiler 18 encounters in a COBOL program instructions to perform an arithmetic operntion-add, subtract, multiply, or divide, for exam-ple-on packed decimal data, it must convert the instructions into one or more machine instructions. In order to cause CPU 15 to perform an arithmetic operstion on the packed decimal data directly, i.e., without converting data typen, compiler 18 generates code to cause CPU 15 to performed the logical functions flowcharted in FIOS. 3 and 2. Regdeter content manapuletions that occur as a consequenice of the functions shown in FIGS. 3 and 8 are symbolically abown in the block diagrams of FIOS 4-7 and 9-12, respectively.

FIG. 4 shows the fusctiom necessary to udd vaiues of two variables, var.A and var.B, of packed BCD data type. It if asumed that var.A is stored to register a (reg.a) 400 and ver. $B$ in atored in readeter $b$ (reg.b) 401, as shown in FIO. A. It is also sanmed that each reglater 400 and 401 ta one 16 -blt-word wide. Each register 400 and 401 comprises a plurality (four) of tields 500 each of which comprises a plurality (four) of bite 501-502. A four-bit field 500 is commonly referred to as a nibble. Fields 500 of regen 400 are sequentially ordered sccording to thetr significance with reapect to each other. Fields 500 of reg.b 401 are correspondingly ordered. Each field S00 stores a BCD digit. To add BCD contents of reges 400 to BCD contents of reg.b 401, the least signifficant btt 508 of each digit fald 500 ts exclusivelyORed with the least significant bit 502 of the corrosponding digtt Eleld 800 of ree.b 402, and the results are stored in correspooding bits 502 of a third regtater 0402 , at step 301 of FIG. 3, m shown ta FIG. A A alx (a binary 0110) is added to cach digit feld 800 of regat 400 with the result being depodted ta rega 400, at step 303 of FIG. 3, and the binary contents of reg-e 400 are then added to the blanary contents of reg.b 401 with the result being depoitted ta reg.b 401, at step 303, se ahowa to FIG. 5. Any carry produced by the operation performed at steps 303 is stored in carry flag 503, as shown in FIG. 8. The value of the leas significant bit 802 of each digit field 800 of reg.b 401 ts then compared with the value of the corresponding bit 502 of reg.e 402, at step 304 of FIO, 3, as shown in FIG. 6. For overy comparison at atep 304 that indicates equality of the com-
pared bits, six is subtracted from the reg.b 401 digit fiel 500 that precedes the digit field 500 corresponding t the compared bits in the ordering of digit fields 500 i : reg.b 401, at step 305 of FIG. 3, as shown in FIGS. and 7. If there is no carry, i.e., if the carry value saver at step 303 is zero, six is subtracted from the most signif icant digit field 500 of reg.b $\mathbf{4 0 1}$, at step 306 of FIG. 3 also as shown in FIGS. 6 and 7. At this point, reg.b 40 holds the BCD sum of the BCD values of the variable previously held by reg.a 400 and reg.b 401.

FIG. 8 shows the functions necessary to subtract thi BCD values of var. $B$ from the BCD values of var.A. I is assumed as before that var. $A$ is stored in reg.a 400 anc var.B is stored in reg.b 401, as shown in FIG. 9. First the least significant bit 502 of each digit field 5000 reg- 400 is exclusively-ORed with the least significan bit 502 of the corresponding digit field 500 of reg.b 401 and the resula are stored in corresponding bits 5010 reg.c 402, at step 801 of FIG. 8, as shown in FIG. 9. Thi binary contents of reg.b 401 are then subtracted-illus tratively through a process of 2's complement addition at is conventional on processors-from the binary con tente of reg.a 400, with the results being deposited ir reg.b 401 and a carry being saved in carry flag 503, a step 802 of FIG. s, as shown in FIG. 10. The value 0 the least dignificant bit 502 of each digit ficld 5000 reg.b 401 is then compared with the value of the corre sponding blt 502 of reg.c 402, at step 803 of FIG. 8, a: shown in FIG. 11. For every comparison at step 80 : that indicates inequality of the compared bits, six is subtracted from the reg.b 401 digit field 500 that pre cedes the digit field 500 corresponding to the comparec bits in the ordering of digit fields 500 in reg.b 401, a step 804 of FIG. 2, as shown in FIGS. 11 and 12. If there ts a carry, hen, if the saved carry value is a one, six it subtracted from the most significant digit field 500 ol reg.b 401, at step 805 of F1O. 8, also ss shown in FIGS 11 and 12. At this point, reg. 401 holds the BCD difference of the BCD vilues of the varitibles previously held by regat 400 and reg.b 401.

Muldiplleation is performed by means of repeatec sdditions and field shifts, while division is performed bs means of repented subtractions and field shifts, as is conventional in procesors.

Code for performing sctivities corresponding to the logical functions of FIOS. 3 and 8 is generated by a compller 18 in compling a packed decimal data ADE or SUBTRACT operation for a procesior 12 that has $n 0$ hardware support for packed decimal data opera. thons. Thl code is flowcharted in FIGS. 13-14 and 15-17.

While compling a souree code program, compiler 18 may encounter an instruction "ADD var.A, var.B", al step 1300, where "var. $\mathrm{A}^{\text {" }}$ is the anme of a first variable of the packed decimal data type and "var. $\mathrm{B}^{\prime \prime}$ is the name of a secood variable of the same type. Compiler 18 knows the data type of the variablea from their declara. tions. In response, complier is first generates object code to perform proceading of the poditive or negative digne of the variables. Sign processing code is conven. tional. For example, compiler 18 generates code ic obtain the signen, at atep 1301, and to compare the signs, at step 1302. Ilustratively, the sign of a variable is stored es the least significant nibble of the one or more memory worde storing the packed decimal variable value, and compller 18 generates code to retrieve anc compare the values of these albbles.

For the case of the two variables being found to have different signs when the object code generated at step 1302 is executed, compiler 18 illustratively generates object code to perform a subtraction of variables of the same sign, as suggested at step 1303, as if the encountered instruction had been "SUB var.A, var.B". This code is shown in FIGS. 15-17 and is discussed below.

For the case of the two variables being found to have the same sign when the object code generated at step 1302 is executed, compiler 18 illustratively generates code to save the sign of var.A, at step 1310.

Compiler 18 then generater code to move the variables into logical or physical regiaters of CPU 15 and to align the variables in the registers by their decimal point position, at steps 1311 and 1312. The code generated to accomplish these functions is likewtiee conventional. For example, compiler 18 socesses the declarations of the variables to determine how many registers are required to hold their cormposite, determines the difference in dectmal potat alignment between the composite and the individual variables, stufte the variables with zeros so that their decimal point poditions become properly aligued with that of the composite, and then generates code to mpve each variable (La, its value) into one or more regdeters. For ease of reference, registers holding var. $A$ are referred to as reg.s an and reginters holding var. B are referred to as regs $\mathrm{b}_{6}$ where if an integer taking values from 0 to one-less then the maximum number of reginters required to bold a varlable: Regas 20 and bo hold the least ifgificent word of the reapective 30 variable value.

To generate object code for the addition operation itself, compiler 18 creates a variable 4 which will be used to count repeated execution of code that follows, and generates code to ret the taltial value to 0 , at step 1313. Next, compiler 18 generates code to perform an exclusive-OR operation on regeyand rez.b, and to mtore the result ta reg.e, sf step 1314. (The perticular segisters a and $b$ involved durtas any oas peen of CPU 18 through this code ars, of courna, determined by the 40 value of I durtas that recmerion)

Complier 18 geveraties code to check the vilue of 4 , at step 1315. For the cme when il in determaned to be noszero, compleer 18 gemerates code to sdd the value of a carry $\mathrm{C}+1$. generated during the precediag execution of the generried code, to conteats of segay and to store the result in regath et ctep 1316.

Compiler 18 thea semerates code, for all values of 1,10 add six to each fred s00 of res.b4 and to store the remitt in reg.by at atep 1317. Compiler 18 mext geomentes code to add coatents of regey to contimata of reg.by and store the result ta reeby tat step 1318 , and to save the value of carry af equerated by chis sdeltion, at step 1315. Next compiler 18 genaratis code to perform an encluatvoOR operation on reeph and regop and store the remilt in reg.ch at etep 1221. Complier is geaserates code to then form a complemeat of the contents of recer sed save
 code to paxt periorin manD operadion between ench
 store the revalts in rege\% et seep 1323 .

Compliar 11 gromerites code to then chick the valio of carry of an enp 1224. For the cate whers the vilue of: $c_{1}$ is zero;'coingiler; 18 senaratee rode to'zed one to reg.ch and wore tho rumit in regon at mop 1324. For all valuce of ch complier 12 generatis code to rotate cone tents of reger ifitit by i places and to tore the remult in reg.ch at step 1326. Compller 18 nent generates code to
subtract contents of res.c, from contents of reg.b/and I store the result in reg.b at step 1327.

Compiler 18 generates code to add contents of reg. to itself, and store the result in reg.c.ch at step 1328. Con piler 18 next generates code to subtract contents 1 reg.ci from reg. $\mathrm{b}_{1}$ and store the result in reg. $\mathrm{b}_{i}$, at ste 1320. Compiler 18 again generates code to check th value of 4 , at step $\mathbf{1 3 3 0}$. For the case when the value 1 I in not the maximum, which is one less than the numb, of registers holding each of the variables A and B , cor piler 18 generites code to increment the value of it one and to return to code generated at step 1314 f another execution of that and subsequent code, at stt 1331.

For the case when the value of $i$ is maximum, $t]$ addition of var.A and var.B is completed, with tl result stored in rega b, Compiler 18 therefore generat conventional code to append the saved positive or neg tive sign to the rerult, at step 1332, and to move tl reanit from regs b, beck into var.B in memory, at st 1333. Compiler 18 then returna, at step 1334, to proce and compile another source program instruction.

When complier 18 encounters an instruction "SL var.A, var. $\mathrm{B}^{\prime \prime}$, directing that the value packed BC data type ver.B be subtracted from the value of di var. $A$, at atep 400 , compler 18 again first general machfoe code to perform procesing of the signs of $t$ variables. The alga procesaing is, again, convention but for the aske of completeness of description, it briefly deacribed. Compiler 18 generates code to obu the igns, at step 1401, and to compare the signs, at st 1402. For the case of the two variables being found have different alga, complier 18 generates code perform an addition of variables of the same sign, a: the encountered linstruction had been "ADD var. var. $\mathrm{B}^{n}$, as raseented at step 1403. This code is shown FIOS. 13-14 and was discresed above.
For the cene of the two vartables being found to ha the same sign when the code of exep 1402 is perform. compler is generites code to obtain the variable I eos, st step 1419, and to compare their absolute valu at step 1411. This teat is done ss if var.A and var. $\mathrm{B} \mathbf{w}$ btary aumbers. Por the case of var.A being greatel magatouda, compiler 18 generntes code to save the s of var.A at step 1412, to move var.A lato reg.s an allm the decimal point poattion, at step 1413, and move var. $B$ tato regs bi and align the decimal $p<$ pocition, at atep 1414. Stepe 1413 and 1414 dupllc stepe 1311 and 1311 of FIG. 13 described above. For cape of ver. 8 belas greater in magraltude, compiler femerates code to mave the oppodte of the sign of vas at atep 1418, to move vars tata regs ay and align dectmal polat poedtion at step 1416, and to move va Into sete by and lifys the decinal poiat ponition, at : 1417.

An elternative to performias stepa 1411
1415-1417 th to always perform stepe 1412-1414 which can the mbeoquently-generated code may re In gemeration of the IO's complement of the des: answer. In that oase, compiler it must generate cod ave carry.vilues that may be produced at steps 1 and s48 end to form a union of these values prio step 142. Compilet 10 also munt generate code, foll tar the below-deieribed code, to check for the complement recult (dilustradively by chooking for al belas prosent in those date fleids of the result thal more sifglificant than any data fielde of the composi
var.A and var.B) and to $10^{\circ}$ s complement such a result to obtain the correct result.

To geaerate object code for the subtraction operation itself, compiler 18 creates a variable $h$ which will be used to count repeated execution of code that follows, and generates object code to set its laitial value to 0 , at step 1418. Next, compiler 18 generates code to perform an exclusive-OR operation on reg.an and reg.bi and to store the result in reg.ch at step 1419.

Compiler 18 then generates code to check the value of $i$, at step 1420. For the case when it determined to be non-zera, compller generates code to add the value of a carry $\mathrm{C}-1$, generated during the preceding recursion through the code, to contents of regay and to store the result in reg.a, at step 1421; to complement contents of reg.b, and store the resuilt in reg.b; at step 1423; to add contents of reg.s to contents of req.bisad atore the result in reg.b at step 1424 ; and to save the value of carry co gencruted by this addition, at step 1425.

For the caso when il is determined to be sero by execution of code generated at step 1420, compiler 18 generates code to subtract contentes of reg.by from contents of regay and store the reault ta reg.b, at step 1427, and to save the value of carry ci senerated by this subtraction, at step 1420.

Next, for all values of $h$, compiler 18 generates code to perform an exclusive-OR operation on reg.b and reg.c and to atore the reault in rezech at step 1429. Compiler 18 generates code to then perform an AND operation between each but the leats sifolficent data field 500 of reg.ciand a one and store the results th reg.ch at step 1430. Compller 18 geaerites code to then check the value of carry $\mathrm{c}_{6}$ at step 1431. For the case where the value of cis one, compiler 18 generates code to add one to reger and store the rerult th regeh at step 1432. For all values of $c_{h}$ compiler generatea code at steps 1433-1439 that duplleater codo senerated at stepe 1326-1333 of FIG. 14 and described above. Compller 18 then returns, at step 1441, 00 process and compile astother source program finstruction.

A progrum, complied es deseribed above, bs rutied for execution On \& procesur 12 that provides no epecial hardware eupport for performatas packed BCD opers. tions. Speed and efficiency of thone operations may be Improved by prowlding hardware support therefor. A suitable modification to the CPU 15 of processor 12 to provide such support is shown surrounded by deahed Hines in FIOS. 11-19. Orly thom conventional portions of the CPU 25 necesary for an understanding of the structure end furction of the modificition are showa in FIGS. 18-15.

CPU 15 ta asumed to be the CPU of a 16-blt processor 12: CPU 18 procemes in paralled the bles of a 16-bit wide word CPU 18 "lllees" each word futo fourbls nibbles CPU is comprises four kdeation stagen, each one of which procemes th peralled four bits of a word and the four staces operate simultaneomaly and perform Bentical operationa. Eech stage tis effoctively a eoparato CPU. Ench stege comprises a conventioal arthmetic and lode witt (ALU) 1800, exch taput port of which in connected by four leads to a difrerent output port of a nibble-wide clice of regdater file 1502 . Reghater flie 1502 holds en array of regitters thcludtag regtiters selected to scrve a regan 400, regab 401, and regen 402. The output port of ALU 1500 is connected by four leads to a bus (aot abown) that, inter alle, connects the outpet port of each ALU 1800 to the taput port of the atsoclated slice of register file 1502. This consection is suggested
in FIGS. 18-19 by the four-lead connection between the output port of each ALU 1500 and the input port of the associated alice of register file 1502.

A carry output terminal of each ALU 1500 is connected by a 1 -bit-wide connection to a carry input terminal of each "subsequent" ALU 1500. A "subsequent" ALU 1500 is one responsible for processing the next most rignificant nibble 500 of a register, whereas a "preceding" ALU 1500 is one responsible for processing the next least significant nibble 500 of each register. The carry output terminal of the last ALU 1500 is connected to a carry latch 1504. Latch 1504 implements carry flag 503.

The modification to the conventional CPU 15 is as follown. The least significant bit 502 leads of each of the two output ports of a slice of register file 1502 are connected to taputs of an exclusive-OR gate 1505. Output of gate 1505 and an ADD/SUB INDICATOR control lead are connected to the lippots of an exciusive-OR gate 1506, whose output is connected to the SELect input of a two-to-one multiplexer 1507. Input ports of multiplexer 1507 are each connected to a different one of two regirters 1508 and 1509. Register 1508 permanently stores a binary 6 value, while register 1509 stores a zero value. Output port of multiplexer 1507 is connected by four leads to an taput port of a two-to-one multiplexer 1510. The other input port and the output port of multiplexer 1510 frtercepts a formerly-direct.connection between an output port of a slice of register file 1502 and an taput port of ALU 1500 .

When compiling programs for a processor 12 having a CPU 15 modified as shown ta FIGS. 18-19, compiler 18 replaces all code gencrated at steps 1321-1329 with a dagle new machine instruction, illustratively named decimal adjust add. Similariy, compiler 18 replaces all code generated at stepi 1429-1436 with a single new mechine fartruction, tluatratively named decimal adjust subtract.

In response to the decimal adjust add instruction, a controtler (not shown) of CPU 15 which directs actions of elements of CPU 15 in performing operations sets to a logical "I" level the ADD/SUBTRACT INDICA. TOR ling, aserts the SEL lead of each multiplexer 1510 to coanect the output port of multiplexer 1507 to the output port of muldiplexer 1510, causes each slice of regirfer file 1502 to outpus contents of the nibble 500 ol res.cy that it ts bolding at the output port connected to maittplexer 1510 and to output at the other output por port contents of the rifble 500 of reg.b, that it is hold. fare, and cases each ALU 1500 to perform a subiract operation. The controiler then causes the output of cach ALU 1000 to be stored ta the aibble 800 of reg.b, held by the coascoted regteter file 1502.

Functions performed by the CPU 15 controller in reaponse to the dectual adjut subtract instruction are Identical to thove performed for the decimal adjust add fintruction, except that the ADD/SUBTRACT INDI. CATOR line tis set to a logical "o" level.

Conteats of fatch 2504 are comblaed with either the "l" add dignal or "O" subtruct signal from ADD/SUB. TRACT INDICATOR line by exclusive-OR gate 1506 Output of gate 1506 is thus " $f^{\prime \prime}$ if there is no carry anc O If there ha a carry on the addition operation, and $1:$ reversed on the subtract operation. A " 1 " output of gate 1506 causes muldiplexer 1507 to channel contents o regirter 1508 to multiplexer 1510. $A^{\text {" }} 0^{\prime \prime \prime}$ output of gati 1506 causes multiplexer 1507 to channel contents 0 regiater 1509 to multiplexer 1510. Assertion of SEL Lin

## 11

of multiplexer 1510 causes it to couple output of multiplexer 1507 to an laput port of ALU 1500.
The least siquificunt bits 502 of two nibbles 500 are combined by exclusive-OR gate 1505. Output of gate 1505 is therefore a " 1 " only if ore inpot is a " 1 " and the other input is a " 0 ". Output of gate 1505 is comblned with the ADD/SUBTRACT INDICATOR line signal by exclasive-OR gate 1506, tin the manner described above for lutch 1504 contents.
Of course, it should be understood that various changes and modifications to the llustrative embodimeat described above will be appareat to those alilled in the art. For example, implementation of the CPU hardware modifications may differ with the design of the particular CPU and the technology used to tumplement the modifications. Ot, the sixes may be sided to an operand before the exclustive-OR of the operands' least rignificent bits is performed. Or, this excluatre-OR operatioa may be replecod by an excluadve-NOR opers. dion or nay set of oae or more operations that produce the seme result: compare the least dignificant bids to deternine and todicite whether they match, Le, aro equal; the subtraction of sices may thea be besed on a compariona tadicuting inequality to the case of eddition and equallty in the care of enbtraction if the fadication of the results of the firt-mentioned comparion produces binary values oppodte to those produced by a posidivo-logic excluadvo-OR operation. Also, the come. parisoas neod mot be performod firat between the leat rignificent blat of the operands and then between the reaulu of thin comparioon and the least ifgaliciant bitas of the result of the curuming or subtricting of the opero ands, but may be performed between the leart sigalncant bita of ooce of the operands and the result of the summing or subtracting of the operande, and then between resulta of chit compariton and the lemat algaificenat blte of the other opernad. Aleo, the tavention many be used to process dildtu coded tan formets having other thea a docimal bese, for expmple, havias a beec of (2a), the difference betas the subutitution of a vilue ( $2^{m-2 i n}$ ) for exch els reod tix the truatrative emboditacat where
 the number of btat ta a deta feld 500 . Furtherwore, the complietion miny be optimizod because the compileer may have caosech informetion avalleble to t -viereble riga tuformetion, for exemplo-ntrom vurible declartr. thoas to that to can avold gemerntiag code for alternas. tives thet th boows will mot exies. Flrally, tastend of a compiler, en fatterpreters or wome ocher entity may be urod to faterpret or otherwive process the fource code inatructione, for exampipio by ciltiag or tavoldas tibrary subrouther or mecroi to expecute the fartructions in the ubove-dencrlbod memper. Accondingly, for purpones of this applicetion, the term "geverattac" of code ticludes withtrit to meentag complitas taterprettus, tavoltas, or calliag. Buch chenget and moditicutions can bo made without deperting frome the epplete and rcope of the to ventioa end without dtadnitebtag tos atteendent adveatages. It is therefore futerded that all such changes and modifications be coverod by the followtas claime.

## What is delmed in:

1. A method of generatiag object-program code from sourco-proprem code, compdetag the stept of: recelvitas cource code eppectryins operation on two opersinde;
seneratitus object code for obraluing two signalh each repreventing a direreant one of the operands and exch havias a plurulity of sequentilly-ordered

Whereby an sbject progeran comprising the generated object code yields the modified resultant sig. nal defintay binery-coded-dectmal values repre. centing the result of the operation on the two oper. ands.
4. The method of claim 3 further comprising the steps of:
generating object code for determining whether execution of the object code for subtracting data field values yielded a carry; and
generating object code for subtracting six from the value of the most-significant data field of the resultant signal if a carry is determined to have been yielded.
5. A method of generating object-program code from 10 source-program code, comprising the steps of:
receiving source code specifying addition of two binary-coded-decimal operands;
identifying two registers each having contents representing a different one of the operands and each 1 including a plurality of sequentially-ordered nibbles each having a plurality of bits including a least-dignificent bit, the values of the plurality of bits of each nibble together representing a binary-coded-decimal value;
generating object code for exclusive-ORing the values toe least-ignificant bits of nibbles of the same order, to obtain to a third register a plurality of bits each havtag an excluaive-OR value corresponding with nibbles of a different order;
generading object code for adding binary values represented by the bits of the two registeri and a value comprising a plurality of sixes, one six for each nibble of one register, to obtain in the one register a remiltart vilue represented by the bits of the one 30 regioter:
generating object code for compartag the values of the least-dignificant bltes of albbles of the ooes regitter each with the value of the corresponding bit of the third reglater; and comparison todicating equality, dix from the value of the nibble of the one register preceding the come. pared values to the order;
whereby en object progmin comprising the genero 40 ated object code to for obtatatig in the mibbles of the one register btasy-coded-dectual vilues representing the cum of the btaxy-coded-decimal values initially beld by the two registers.
6. The method of claim 8 further compriating the stepe if of:
generatins object code for determining whether execution of the object code for addias ylelded a carry; and
gencratias object code for eubtracting sux trom the 50 moet-ifgnificent nibble of the ove regiater if a carry Is determined not to have been yielded.
7. A method of generatios object-pro gram coda trom source-progrem code, compretas the etepe of:
receivtas cource code spectiytus subtriction of two 35 binary-coded-dectanal operandes,
identifytas two refisters each having contents representiag a dirfarent oas of the operands and ach includins a plurality of sequentally-ordered nibbles each having a plurality of bite lacluding a leaut-dgmificant 6 th, the velues of the plurality of bits of each ribble together representing a blasy-coded-dectmal vilue;
generating object code for exclraive-ORing the valwes of the least-dgrificint bits of nibbles of the 65 same order, to obtaia ti a third regarter a plurality of bitis each havias an exclusive-OR value corresponding with nibbles of a different order;
generating object code for subtracting binary values represented by the bits of the two registers, to obtain in one register a resultant value represented by the bits of the one register;
generating object code for comparing the values of the least-significant bits of nibbles of the one register each with the value of the corresponding bit of the third register; and
generating object code for subtracting, for every comparison indicating inequality, six from the value of the nibble of the one register preceding the compared values in the order;
whereby an object program comprising the generated object code is for obtaining in the nibbles of the one register binary-coded-decimal values representing the difference of the binary-coded-decimal values initially held by the two registers.
8. The method of claim 7 further comprising the steps of:
generating object code for determining whether execution of the object code for subtracting the two registers' binary values yielded a carry; and
generating object code for subtracting six from the motteignificant nibble of the one register if a carry in determined to have been yielded.
9. A method of allocating a resource to a resource user comprising the steps of:
representing two portions of the resource each by a different dgnal, each signal having a plurality of sequentally-ordered data fields each of which includes a plurality of bler, tacluding a least-significant bith, defining a binary-coded-decimal value;
sumantig binary values of the data flelds of the same sequential order and a value comprising a plurality of sixes, ome six for date fields of the same sequential order, to get a resultant sigual defining a value representiag the remult of the summing and having a plurality of deta fields orderied correspondingly to the dita ficids of the sepresenting signals and each including a plurality of bita including a leastsignificant bit;
exanining values of least-dignificant bits of data fields of the eame sequential order of the representing signals and the remultant adgral;
for each examinatiou of leantrignificant bit values that include an odd number of zeros, subtracting six trom the value of the data field of the resultant rignal preceding the compared values in the sequential order of data fields, to obtain a modified remultant signali asd
allocating to the rear a portion of the resource represented by blaary-coded-decimal values defined by the modified retultant sigual.
10. The method of claim 9 further comprising the steps of:
determining whether the step of summing yielded a carry; and
subtracting six from the value of the mostaignificant data field of the resultant signal if a carry is determined sot to have been yiolded.
11. A method of illocating a resource to a resource meer comprising the steps of:
sepresenting two portlons of the resource each by a different signal, each sigan having a plurality of sequentially-ordered datia fields each of which includes a plurality of bites including a least-signifi. cant blt defining a binary-coded-decimal value;
subtracting binary values of the data fields of the same sequential order, to get a resultant signal defining a value representing the result of the subtracting and having a plurality of data fields ordered correspondingly to the data fields of the representing signals and each including a plurality of bits including a least-significant bit;
examining values of least-significant bits of data fields of the same sequential order of the representing signals and the resultant signal:
for each examination of least-significant bit values that include an odd number of ones, subtracting six from the value of the data field of the resultant signal preceding the compared values in the sequential order of data fields, to obtain a modified is resultant signal; and
allocating to the meer a portion of the resource represented by blany-coded-dectmal values defined by the modified resultant signel.
12. The method of claim 11 further comprising the 20 steps of:
determintag whether the step of subtracting data field values ylelded a carry; and
subtructing six from the value of the mont-significunt data field of the resultant aignal if a carry is deter- 25 mined to have been yielded.
13. A method of allocating a resource to a resource user comprialig the stepe of:
representing two portions of the resource each by contents of a different register, each register including a plurality of sequentially-ordered sibbles each having a plurality of bits, lacluding a least-aignificant bit, and defting a btnary-coded-decimal value;
exclusive-ORing the values of the leant-dgrificant 35 bita of nibbles of the same order, to obtain in a third register a plurality of bits each having an axclusiveOR value correaponding with aibbles of a different order,
adding binary viluea represented by the blts of the 40 two regdeters and a value comprising a plurality of sures, ove six for each nibble of ooe reghater, to obtain th the oce register a resultant value reprosented by the bits of the one regeter;
comparing the values of the least-dgrificant bits of as nibbles of the oae regtster each with the value of the corresponding bit of the third register;
for every comparion ladicatias equality, subtracting sta from the value of the ribbie of the one register preceding the compared valves tin the order; and
allocating to the maer a portion of the resource represented by binary-ooded-dectmal valver contained to the ove segdeter.
14. The method of clatim 13 further comprtatis the stepe of:
determiniag whether the step of addtas yielded a carry; and
subtracting six from the vilue of the mott-ifgificant uibble of the oce reghter if a carry in determined not to have been yielded.
15. A method of allocating a secource to a rewource user compristag the etepe of
representing two portions of the resource each by contents of a different regtoter, each register tocluding a plurallity of sequentially-ordered nibblea each haviag a plurallty of bitu, including a leant-significaat bit, and defindigg a blamy-coded-decimal value;

 . stepe of
determiming whether the step of subtracting the tu regdeters' binary values yielded a carry; and
subtracting six from the value of the mostsignifica nibble of the one register if a carry is determined have beea ylelded.
17. A method of operating a digital computer $f_{i}$ combining two received digital signals, each having plurality of sequentially-ordered data fields each which includes mbits including a least-significant $t$ and a value defined by the bits representing a binar coded base-(2a) value, mand $a$ being positive intege euch that $2^{m}>2 a$, the method comprising the steps o
summing binary valuea of the data fields of the san sequental order and a plurality of values eac equal to $\left(2^{m}-2 n\right)$, one vilue for data fields of $t 1$ same sequental order, to get a resultant sign defining a value representing the result of the sur ming and having a plurality of data fields ordert correspoadingly to the data fields of the receiv1 dignalis and each including m bits including a lea signifteant ble:
comparing values of least-significant bits of data the same sequentila order of the received signs and the resultant signel; and
for each compartion of least-rignificant bit values th toclude an odd number of zeros, subtracting a val equal to $(2 m-2 n)$ from the value of the data field the remultant signal preceding the compared valu ta the eequential order of data fields, to obtain the data felds of the recultant signal binary-codi beeo-(2n) viluet representing the combination the recetved binary-coded beso-( 2 n ) values.
18. The method of clatm 17 wherein m equals fo and a equala five.
13. The method of claim 17 further comprising t. stepe of
determintas whether the step of summing yielded carry; and
suberncting a value equal to $\left(2^{m}-2 n\right)$ from the val of the mot-dgrificant data fieid the resultant sigr If a carry la determined not to have been yielde
20. A method of operating a digital computer it combialag two recelved digital dignale, each having plurality of sequentially-ordered data fields each which includes m bits lacluding a least-significant and a value defined by the bits representing a binal
coded base-( 2 n ) value, $m$ and $n$ being positive integers such that $2 m>2 n$, the method comprising the steps of: binary values of the data fields of the same sequential order, to get a resultant signal defining a value representing the result of the subtracting and having a plurality of data fields ordered correspondingly to the data fields of the received signals and each including $m$ bits including a least-significant bit;
comparing values of least-significant bits of data fields of the rame requential order of the received signals and the resultant signal; and
for each comparison of lesst-iignificant bit values that include an odd number of ones, subtrncting a value equal to $\left(2^{m}-2 n\right)$ from the value of the datia field of the resultant signal preceding the compared values in the sequential order of data fields, to obtrinio in the data fields of the resultant dignal binary-coded bace-(2n) values representing the combination of the recelved binary-coded base-(2n) values.
21. The method of claim 20 wherelin mequals four and $n$ equals five.
22. The method of claim 20 further compriating the steps of:
determining whether the step of subtracting data field 25 values yielded a carry; and
subtracting a value equal to $\left(2^{m}-2 n\right.$ ) from the value of the mont-dgnificant deta fleld of the reaultent signal if a carry is determined to have been yielded.
23. A method of operating a digtal computer to combine contents of two registers, exch including a plurality of sequentially-ordered nibbles exch having a plurality of bits including a leant-dgaificant bth, the values of the plurality of bita of each albble together representing a binary-coded-decimal value, the method comprising the 35 steps of:
excluave-ORing the values of the least-rignificant bits of nilblee of the sume order, to obtrin ta a thurd register a plurality of btes ench havitg en excluaiveOR value correeponding with aibbles of a different 40 order;
adding btarary valveen represented by the bitu of the two registers and a value comprtatig a plurality of sixes, ove sux for each mibble of one regiteter, to obtata to the ooe regiterer a recultant value represented by the bita of the one regtater;
compertas the values of the leat-dicaificint bita of nibbles of the one reghter each with the value of the corresponding bit of the third regitter; and
for every cocmperisoa bodicating equelity, zabernettag sux trom the value of the nibble of the ose segitter preceding the compered valuee in the order, to obstin tai the albblet of the oose regideter binary-cod-ed-dectonal valuee repreventiag the sum of the bf-arry-ooded-decimal valuen fuititilly beld by the two registers.
24. The method of olam 23 flurther compritang the stepe of
detersmining whether the atop of addtus yidided a carry; and
ruberncting six from the value of the mox-dignificant nibble of the coe requitar if a carry is determined not to have been ylelided.
25. A method of operating a digital computers to 00 mbine contentu of two rogeteters, ench facludtag a pluralty 65 of sequentlally-ordered aibbiles ench haviag a plurallty of bits locludiag a lomet-dgnificeat ble, the valuio of the plurality of biti of each aibble together represeanting a
binary-coded-decimal value, the method comprising the steps of:
exclusive-ORing the values of the least-significant bits of nibbles of the same order, to obtain in a third register a plurality of bits each having an exclusive. OR value corresponding with data fields of a differeat order;
subtracting binary values represented by the bits of the two registers, to obtain in one register a resultant value represented by the bits of the one regis. ter;
comparing the values of the least-significant bits of nibbles of the one register each with the value of the corresponding bit of the third register; and
for every comparison indicating inequality, subtracting a six from the value of the nibble of the one register preceding the compared values in the order, to obtain in the nibbles of the one register binary-coded-decimal values representing the difference of the binary-coded-decimal values initially held by the two registers.
26. The method of clalm 25 further comprising the steps of:
determining whether the step of subiracting the two registeri' binary values yielded a carry; and
rubtracting dix from the value of the most-significant nibble of the one reginter if a carry is determined to have been yielded.
27. An arrangement for generating object-program code from source-program code, comprising:
first means, responaive to receipt of source code specifyling an operation on two binary-coded-decimal operinds, for generating object code for obtaining two signals each representing a different one of the operande and each having a plurality of sequential-ly-ordered data fiolds esch of. which includes a pluralty of biter locluding a least-significant bit,

- defining a binary-coded-decimal value;
scond means, cooperative with the first means, for generating object code for summing binary values of the data feelds of the same sequential order and a value comprising a plurnility of sixes, one six for data fielde of the same sequential order, to get a remulant aignal defining a value representing the result of the summing and having a plurality of data fielde ordered correspondingly to the data fields of the obtained algnals and each including a plurality of bits including a least-dignificant bit;
third meass, cooperadive with the first and the second means, for generating object code for examining values of lemet-dignificant bita of data fields of the sames sequential order of the obtrined signals and the remultant signal; and
fourth meense, cooperntive with the chird means, for gemerating object code for subtrecting, for each examhanton of least-ignificent values that include san odd number of zerce, six from the value of the deta field of the resultant alignal preceding the compared values ta the sequential order of data fields, to obtuin a modified resultant signal;
whereby an object program comprising the object code generatod by the first through fourth means ylalde the modified renultant rignal defining binary-coded-dectmal values representing the result of the operation on the two operands.

23. The arrangement of claim 27 Aurther comprising:
afth meante, cooperative with the cooond means, for generating object code for determining whether
execution of the summing object code yielded a carry; and
sixth means, cooperative with the fifth means, for generating object code for subtracting six from the value of the most-significant data field of the resultant signal if a carry is determined not to have been yielded.
24. An arrangement for generating object-program code from source-program code, comprising:
first means, responsive to receipt of source code spec- 10 ifying an operation on two binary-coded-decimal operands, for generating object code for obtaining two signals each representing a different one of the operands and each having a plurality of sequential-ly-ordered data fields each of which includes a is plurality of bith, including a least-aignificant bit, defining a binary-coded-decimal values
second means, cooperative with the first mean for generating object code for subtracting binary values of the data fields of the same sequential order, to get a remuleant idgal defining a value representing the result of the subtracting and having a plorality of data fields ordered correapondingly to the data fields of the obtained signats and each including a plurality of bits including a least-lignificant 2 bit;
third means, cooperative with the first and the second meana, for generating object code for examining values of least-dgrificant bits of data fielde of the same sequential order of the obtained aigmals and the resultant signal; and
fourth meame, cooperative with the third means, for generating object code for subtracting. for each examination of least-ignificant bit values that include an odd number of onea, six from the value of the deta field of the resultant eignal preceding the compared values in the sequential order of data fields, to obtata a modified remulant signal;
whereby en object program comprising the object code generated by the firk through fourth means yielde the modibed remiltant edgal deflatag binary-code-dectmal values representing the restite of the operation on the two operaods.
25. The arrangement of ciatio 29 further compridag:
finh meam, cooperative with the second meane, for generuting object code for determining whether execution of the object code for subtracting deta field values ylelded a carry; tad
sixth menos, cooperative with the fith means, for generating object code for subtracting six from the value of the mont-dgificant date field of the recultant dignal if a carry is determined to have been yielded.
26. As arrangement for geseratins objeot-pirogram code from source-program code, compriatagy
first meane, responetve to reotpt of cource code speoifying addition of two thenty-coded-deotmal operande, for ldenetilying two reghters emph havtas conteats represattas a differeat oue of the opers ands and esch fuciudins a plurality of sequentillyordered mblobies each havios a plurality of blas includias a leastelgaificunt bth the values of the plurality of bits of each aibble together repreecating a binary-coded-decimal value;
second means, cooperative with the first means, for 65 generading object code for excluaive-ORing the values of leantedgrifiomat blts of aibbles of the same order, to obtala ta a thind reglater a plurallity of bita
fourth moans, cooperatlve with the first and the se ond means, for generating object code for compa tug the values of the least-digulficant bits of nibbl of the one segister each with the value of the con sponding bit of the cthird regiater; and
fith means, cooperative with the fourth means, geverating object code for subtracting, for eve comparison indicating inequality, six from $t$ value of the aibble of the one reglater preceding : compared values in the order;
whereby an object program comprising the object code generated by the first through fifth means is for obtaining in the nibbles of the one register bi-nary-coded-decima! values representing the difference of the binary-coded-decimal values initially held by the two registers.
27. The arrangement of claim 33 further comprising: sixth means, cooperative with the third means, for generating object code for determining whether execution of the code for subtracting the two registers' binary values yielded a carry; and
seventh means, cooperative with the sixth means, for generating object code for subtracting six from the value of the most-significant nibble of the one register if a carry is determined to have been yielded.
28. A system for allocating a resource to a resource user, comprising:
first means, for representing two portions of the resource each by a different gignal, each signal having a plurality of sequentially-ordered data fields each of which includes a plurality of bits, including a least-significant bit, defining a binary-coded-decimal value;
second means, cooperative with the first means, for summing binary values of the data flelds of the 25 same sequential order and a value comprising a plurality of sixes, one six for data fields of the same sequential order, to get a resultant signal defining a value representing the the result of the summing and having a plurality of data ficlds ordered correspondingly to the data fields of the representing signals and each including a plurality of bits including a least-significant bit;
third means, cooperative with the first and second means for examining values of least-significant bits 35 of data fields of the same sequential order of the representing aignals and the resultant signal;
fourth means, cooperative with the third means, for subtracting. for each examination of leat-igigifcant bit values that include an odd number of zero, sin from the value of the data feld of the remultant signal preceding the compared valuea in the sequential order of data fielde, to obtain a modified resultant aiganl; and
fint means, cooperative with the fourth means, for 45 allocating to the user a portion of the resource represented by blary-coded-decimal values dofined by the modified resultant signal.
29. The system of clalm 35 further compriding:
sixth means, cooperadive with the second means, for so determining whether the summing performed by the second means yleided a carry; and
seventh means, cooperative with the sixth means, for subtracting six from the value of the moet-ignificant data field of the resultant signal, If a carry is determined not to have been ylelded.
30. A syatem for allocating a resource to a resourca uscr, comprising:
first means, for representing two portions of the rosource each by a differeat signil, each signal hav. 60 ing a plurality of sequentially-ordered data fields esch of which inclodes a plurallity of bles, including a least significant bit, defining a binary-coded-decimal values
second means, cooperative with the firat means, for 65 subtracting binary values of the data fields of the same sequential order, to get a resultant signal defining a value representing the the result of the
third means, cooperative with the first and second means, for examining values of least-significant bits of data fields of the same sequential order of the representing signals and the resultant signal;
fourth means, cooperative with the third means, for subtracting, for each examination of least-significant bit values that include an odd number of ones, six from the value of the data field of the resultant signal preceding the compared values in the sequential order of data fields, to obtain a modified resultant signal; and
firth means, cooperative with the fourth means, for allocating to the user a portion of the resource represented by binary-coded-decimal values defined by the modified resultant signal.
31. The syatem of claim 37 further comprising:
sixth means, cooperative with the second means, for determining whether the subtracting performed by the second means yielded a carry; and
seventh means, cooperative with the sixth means, for subtracting six from the value of the most-significant data field of the resultant signal, if a carry is determined to have been yielded.
32. A syatem for allocating a resource to a resource user, comprising:

## three registers;

first meanis, for representing two portions of the resource each by contents of a different register, each register including a plurality of sequentiallyordered nibbles each having a plurality of bits, including a least significant bit, defining a binary-coded-decimal value;
second means, cooperative with the first means, for exclusive-ORing the values of the least-significant bits of nibbles of the same order, to obtain in a third register a plurallity of bits each having an exclusive. OR value cor spoading with nibbles of a different order;
thind meass, cooperative with the first means, for adding blanry values represented by the bits of the two reghters and a value comprising a plurality of sixes, one dx for each nibble of one register, to cbtala in the one reginter a resultant value represented by the bits of the one register;
fourth means, cooperative with the first and second mean, for comparing the values of the least-significuat bitu of nibbles of the one register each with the value of the corresponding bit of the third register;
fith means, cooperative with the fourth means, for subtracting, for every comparison indicating equaltry, adx from the value of the nibble of the one register preceding the compared values in the order; and
sixth means, ccoperative with the fifth means, for allocating to the user a portion of the resource represented by binary-coded-decimal values contalined in the one register.
40. The syrtem of claim 39 further comprising:
seventh means, cooperative with the third means, for determining whether the adding performed by the third means ylelded a carry; and
eighth means, cooperative with the seventh means, for subtructing six from the value of the mostsig-
nificant nibble of the one register, if a carry is determined not to have been yielded.
41. A system for allocating a resource to a resource user, comprising:
three registers;
first means, for representing two portions of the resource each by contents of a different register, each register including a plurality of sequentiallyordered nibbles each having a plurality of bits, including a jeast significant bit, defining a binary-codeddecimal value;
second means, cooperative with the first means, for exclusive-ORing the values of the leat-igignificant bits of nibbles of the same order, to obtain in a third register a plurality of bits each having an excluaiveOR value corresponding with nibbles of a different order;
third means, cooperntive with the firnt meanis, for subtracting binary values represented by the bits of the two registert, to obtain in the one reglater a resultant value represented by the bits of the one register;
fourth means, cooperutive with the first and second means, for compartag the values of the leastrignithcant biss of nibblet of the one register each with the value of the corresponding bit of the third register;
fith meant, cooperative with the fourth means, for subtructing for overy comparison indicating inequallty, six from the value of the ribble of the one register preceding the compered values ta the order; and
sixth meast, cocperative with the fith means, for allocating to the user a portion of the resource represented by blary-coded-decimal valves con-

42. The symtem of ciefm 41 surther compridig:
seventh mean; cooperntive with the third meane, for determinias whether the subtraction performed by the third means yielded a carys and
eighth meass, cooperndive with the seventh meams gubtracting dix trom the vilue of the moterignis cant nibble of the ove regheter, if a carry in determined to have been gielded.
43. A dighal competar arranged for combtatas two signals, comprintare:
a pluralify of mease for representing dignits, each of the two slgmals being represented by differeat dis: nal-reprecentine meam, each of the two signale' representios meana having a plurallty of requeptini. iy-ordered dats belds each of which facludes on bits inclediag a lano-diputiceme bli and a valua defined by the bite repreventing a blatry-coded bae-(2a) value, in and in betas poddive tategers guch that $2^{m}>2 n_{4}$
a plurnlity of means for repreeenting signals, each of the two signale betas riepresented by different sig. ral-representing mean, each" of the two signals' a repreceating metias having a plurality of sequential-
ly-ordered dete Eelds each of which includes m bits moludias a least-lignificant bit and a value defined by the bits repsesentiog a binary-coded base-(2n) valua, in and in betag poditive integers such that $2^{m}>2 m$
mearai-cooperative with the iggal-representing : meani, for zabtractiag blany values of the data Belds of the mame sequential order of the two sigmalo sepresenting means, to obtain a resultant siganal sepremented by one of the plurality of signalreprecenting means, the resultant signal-represent. for meand defining a value representing the resulı of the auberactias and having a plurality of data

- Belds ardesed correspondingly to data fields of the two sifnale' reprewenting meana and each including en bitis inciudtas a leat-lignificant bit;
meane' cooperative with the dgnal-representing menne, for compertos values of least-significani stes of dati fielde of the same sequential order of the two sifmald representing means and the resul. tant sigmoreppresenting means; and
meen, copperattve with the comparing means anc the domi-reprecentus means, for subtracting. fo cach compariso of loent-ignificant bit values in cludtas an odd sumber of ones, a value equal to $(2 \pi-2 N)$ trom the value of the data field of thi semitant dignal-represeating means preceding th. compered values la the equential order of dat foild, to obtata to the data fields of the resultan digal-represeating means blanry-coded base-( $2 n$ valuea represeating the combination of the binary coded bue-(2a) values faitially represented by th in two dignle' representas means.

47. The computer of claim 46 wherein m equals fourand $n$ equals five.
48. The computer of claim 46 further comprising:
means, cooperative with the data field binary value subtracting means, for determining whether the subtracting of data field values yielded a carry; and
means, cooperative with the determining means and signal-representing means, for subtracting a value equal to $\left(2^{m}-2 n\right)$ from the value of the most-significant data field resultant signal-representing means, if a carry is determined to have been yielded.
49. A programmed digital computer comprising:
a first and a second register, each including a plurality of sequentially-ordered nibbles each having a plurality of bits including a least-significant bit, the values of the plurality of bits of each nibble together representing a binary-coded-dectual value;
a third register;
means, cooperative with the three regiaters, for exclu-sive-ORing the values of the leat-dignificant bits of nibblea of the same order of the first and the second register, to obtain th the third register a plurallity of bits each having an excluaive-OR value corresponding with nibbles of a different order;
means, cooperative with the flut and the second register, for adding binary vilues represented by the bles of the two reginters and a value comprising a plurality of alres, one ix for each nibble of one register, to obtatn in the first register a remultant value represeated by the bits of the firt register;
means, cooperative with the firt and the third register, for comparing the values of the least-significant bits of nibbles of the first regleter each with the value of the corresponding bit of the third register; and
means, cooperative with the compartag means and with the linxt regester, for gubtracting, for every comparioon fodicating equallty, ifx from the value of the nibble of the flrst regtater precedips the compered values in the order, to obtain to the nibbles of tha first regheter binary-coded-decimal vab: ves reproventing the sum of the binery-coded-deeb- is mal vilues fattially beld by the first and the second regheter.
50. The computer of chinn tionther compriting:
means, cooperative with the adding means, for determining whether the addition yielded a carry; and
means, cooperative with the first register and with determining means, for subtracting six from the value of the most-significant nibble of the first register if a carry is determined not to have been yielded.
51. A programmed digital computer comprising:
a first and a second register, each including a plurality of sequentially-ordered nibbles each having a plurality of bits including a least-significant bit, the values of the plurality of bits of each nibble together representing a binary-coded-decimal value;
a third register;
means, cooperative with the three registers, for exclu-sive-ORing the values of the least-significant bits of nibbles of the same order of the first and the second register, to obtain in the third register a plurality of bits each having an exclusive-OR value corresponding with nibbles of a different order;
means, cooperative with the first and the second register, for subtracting binary values represented by the blas of the two registers, to obtain in the first register a resultant value represented by the bits of the flut register;
meane, cooperative with the flrst and the third register, for comparing the values of the least-significant bits of nibbles of the first register each with the value of the corresponding bit of the third register; and
means, cooperative with the comparing means and with the firt register, for subtracting, for every comparson Indicating Inequality, six from the value of the aibble of the first register preceding the compared values in the order, to obtain in the nibbles of the first regiater binary-coded-decimal values sepresenting the difference of the binarycodeddecimal values initially beld by the first and the second registers.
52. The computer of claim 81 further comprising:
mans, cooperative with the register binary value subtracting means, for determining whether the mbirsction yielded a carry; and
means, cooperative with the first register and with the determiaing means, for subtracting six from the value of the mont-ligulficant albble of the first regtater if a carry is determined have been yielded.
