

627753  
FORM 1  
REGULATION 9

COMMONWEALTH OF AUSTRALIA

PATENTS ACT 1952-1973

APPLICATION FOR A PATENT

I/We TANDEM COMPUTERS INCORPORATED

of 19333 Vallco Parkway, Cupertino, CALIFORNIA 95014, U.S.A.

hereby apply for the grant of a Patent for an invention entitled:

A METHOD AND APPARATUS FOR MODIFYING  
MICRO-INSTRUCTIONS USING A MACRO-INSTRUCTION PIPELINE

which is described in the accompanying complete specification. This Application is a Convention Application and is based on the Application(s) numbered: 036,726 for a Patent or similar protection made in U.S.A. on 10 April 1987.

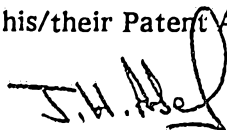
My/Our address for service is:

GRIFFITH HASSEL & FRAZER  
71 YORK STREET  
SYDNEY N.S.W. 2000  
AUSTRALIA

DATED this 8th day of April, 1988.

TANDEM COMPUTERS INCORPORATED

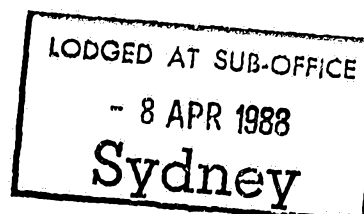
By his/their Patent Attorneys



GRIFFITH HASSEL & FRAZER

TO: THE COMMISSIONER OF PATENTS  
COMMONWEALTH OF AUSTRALIA

1853A:rk



APPLICATION ACCEPTED AND AMENDMENTS

ALLOWED 14.12.90

PATENT DECLARATION FORM (CONVENTION)  
COMMONWEALTH OF AUSTRALIA  
Patents Act 1952

10577-97

Regulation  
12 (2)

DECLARATION IN SUPPORT OF A CONVENTION APPLICATION  
FOR A PATENT

To be signed by the applicant(s) or in the case of a body corporate to be  
signed by a person authorised by the body corporate.

In support of the Convention application made for a patent for an invention entitled

(a) Insert title  
of invention.

(a) ..... A METHOD AND APPARATUS FOR MODIFYING MICRO-INSTRUCTIONS  
USING A MACRO-INSTRUCTION PIPELINE

(b) Insert full  
name(s) of  
declarant(s).

(b) I, Anthony T. Cascio

(c) Insert  
address(es) of  
declarant(s).

~~xxxx~~ (b) C/o Tandem Computers Incorporated

of (c) 19333 Valico Parkway, Cupertino, California 95014

do solemnly and sincerely declare as follows:-

1. ~~I am/We are the applicant(s) for the patent~~

(OR, IN THE CASE OF AN APPLICATION BY A BODY CORPORATE.)

1. I am ~~We are~~ authorised by TANDEM COMPUTERS INCORPORATED

..... the applicant for the patent to make this declaration on its behalf.

2. The basic application(s) as defined by Section 141 of the Act was/were made in the following  
country or countries on the following date(s) namely:-

(d) Insert  
country in  
which basic  
application(s)  
was/were  
filed.  
(e) Insert date  
of basic  
application(s).  
(f) Insert full  
names of basic  
applicant(s).

in (d) the United States of America on (e) 10 April 1987

by (f) Daniel E. Lenoski

in (d) ..... on (e) .....

by (f) .....

in (d) ..... on (e) .....

by (f) .....

3. ~~I am/We are the actual inventor(s) of the invention referred to in the basic application.~~

(OR, WHERE A PERSON OTHER THAN THE INVENTOR IS THE APPLICANT)

(g) Insert full  
name(s) of  
actual  
inventor(s)  
(h) Insert  
address(es) of  
actual  
inventor(s).

3. (g) Daniel E. Lenoski

of (h) 255 South Rengstorff Avenue, #44, Mountain View,  
California 94040

is/are the actual inventor(s) of the invention and the facts upon which the applicant(s) is/are entitled to  
make the application are as follows:

(i) Set out how  
applicant(s)  
derive(s) title  
from actual  
inventor(s)  
i.e., assignee of  
the invention  
from the actual  
inventor(s).  
Attestation or  
legalization  
not required.

(i) assignment dated 8 April 1987 whereby the applicant  
is the assignee of the said invention from the said  
inventor.

4. The basic application(s) referred to in paragraph 2 of this Declaration was/were the first  
application(s) made in a Convention country in respect of the invention the subject of the application.

To:

The Commissioner of Patents

Declared at Cupertino this 12<sup>th</sup> day of January 1988

California

Tandem Computers Incorporated

By:  Signature of Declarant(s)  
Anthony T. Cascio, Assistant Secretary

---

**(12) PATENT ABRIDGMENT      (11) Document No. AU-B-14435/88**  
**(19) AUSTRALIAN PATENT OFFICE      (10) Acceptance No. 607753**

---

(54) Title  
A METHOD AND APPARATUS FOR MODIFYING MICRO-INSTRUCTIONS USING A  
MACRO-INSTRUCTION PIPELINE

International Patent Classification(s)  
(51)<sup>4</sup> G06F 009/38

(21) Application No. : 14435/88

(22) Application Date : 08.04.88

(30) Priority Data

(31) Number	(32) Date	(33) Country
036726	10.04.87	US UNITED STATES OF AMERICA

(43) Publication Date : 13.10.88

(44) Publication Date of Accepted Application : 14.03.91

(71) Applicant(s)  
TANDEM COMPUTERS INCORPORATED

(72) Inventor(s)  
DANIEL E. LENOSKI

(74) Attorney or Agent  
GRIFFITH HACK & CO., GPO Box 4164, SYDNEY NSW 2001

(57) Claim

1. A method for executing instructions in a controller comprising the steps of:

    fetching with said controller a first macro-instruction and a next macro-instruction;

    executing with said controller concurrently during a single clock cycle first and second operations, said first operation being used to carry out said first macro-instruction and said second operation being performed to carry out said next macro-instruction;

    decoding with said controller said next macro-instruction to determine whether less than a predetermined number of operations are required; and

    using in said controller the result of said second operation during execution of said next macro-instruction if said next macro-instruction has a number of operations less than said predetermined number and otherwise re-executing said second operation.

607753

COMMONWEALTH OF AUSTRALIA

PATENTS ACT 1952

Form 10

**COMPLETE SPECIFICATION**

**FOR OFFICE USE**

Short Title:

Int. Cl:

Application Number:  
Lodged:

This document contains the  
amendments made under  
Section 49 and is correct for  
printing

Complete Specification-Lodged:  
Accepted:  
Lapsed:  
Published:

Priority:

Related Art:

---

**TO BE COMPLETED BY APPLICANT**

Name of Applicant: TANDEM COMPUTERS INCORPORATED

Address of Applicant: 19333 Vallco Parkway, Cupertino,  
CALIFORNIA 95014, U.S.A.

Actual Inventor: Daniel E. Lenoski

Address for Service: GRIFFITH HASSEL & FRAZER  
71 YORK STREET  
SYDNEY NSW 2000  
AUSTRALIA

Complete Specification for the invention entitled:

**A METHOD AND APPARATUS FOR MODIFYING  
MICRO-INSTRUCTIONS USING A  
MACRO-INSTRUCTION PIPELINE**

The following statement is a full description of this invention,  
including the best method of performing it known to me/us:-  
1853A:rk

A METHOD AND APPARATUS FOR CODIFYING MICRO-  
INSTRUCTIONS USING A MACRO-INSTRUCTION PIPELINE

5

BACKGROUND

The present invention relates to the execution of macro instructions by a central processing unit utilizing sequences of microcode instructions.

10 In a typical modern computer, a program is executed by fetching an instruction from memory and placing it in an instruction register. The instruction is then decoded to point to a starting address or series of addresses in a microcode memory. The micro-  
15 code memory provides the operations which make up the instruction. The various operations from the microcode memory are sequentially placed into a micro instruction register where they are decoded to produce control signals for executing the operations. These control  
20 signals may enable an access of memory, the placement of operands into an arithmetic logic unit, the combination of operands in an arithmetic logic unit, etc. After all the microcode operations for a particular macro-instruction have been executed, a new macro-  
25 instruction is fetched from memory and the process is repeated.

Once the macro-instruction has been decoded, there is typically no interaction between the micro coded operations and the macro-instruction except for  
30 instances in which the macro-instruction includes a data operand or a register specifier for a data operand.

In efforts to speed computer operation, attempts have been made to shorten the number of clock  
35 cycles required for the macro instructions. One method of doing this involves performing redundant microcode operations and storing the results of these operations

in separate registers where necessary. The next macro-instruction ca. then be decoded to determine whether it requires these operations. If it does, the precomputed results can be used. If not, the result of the redundant operation is thrown out. Unfortunately, this method requires a significant amount of additional hardware and often results in wasted operations. This type of scheme is employed in the TXP and VLX processors manufactured by Tandem Computers, Inc.

Another method involves processing multiple microcode instructions at one time to do some operations not requiring the ALU in parallel with ALU operations. The advantages of increased speed and simplified control logic are balanced by the disadvantage of requiring more hardware and making microcode branches slower.

Another method involves simply hardwiring certain macro-instruction operations so that microcode does not have to be accessed at all for such operations. The obvious disadvantage of this method is that the hardwired circuit becomes dedicated to that function and can't be used for other purposes.

U. S. Patent No. 4,312,034 describes yet another method for reducing the amount of time required to execute a macro-instruction. Referring to Fig. 15 of that patent, a macro-instruction register (IRD) and a ROM output register (microcode) are factored into a ROM address whose outputs control an ALU and condition codes. Thus, the macro-instruction itself is used to control the ALU and condition codes instead of relying on the microcode instructions entirely. Thus, for example, to do an add or subtract operation the microcode would simply do the same fetch operation with the controller looking directly to the macro-instruction to determine whether to add or subtract.

### SUMMARY OF THE INVENTION

According to a first aspect of the present invention there is provided a method for executing instructions in a controller comprising the steps of:

5        fetching with said controller a first  
macro-instruction and a next macro-instruction;  
      executing with said controller concurrently during a  
single clock cycle first and second operations, said first  
operation being used to carry out said first  
10       macro-instruction and said second operation being  
performed to carry out said next macro-instruction;  
      decoding with said controller said next  
macro-instruction to determine whether less than a  
predetermined number of operations are required; and  
15       using in said controller the result of said second  
operation during execution of said next macro-instruction  
if said next macro-instruction has a number of operations  
less than said predetermined number and otherwise  
re-executing said second operation.

20       Preferably only one of said first and second  
operations is a memory access operation. Alternatively  
said first operation may be incrementing a program counter  
and said second operation may be a data operand fetch.

25       The method preferably comprises the further steps of:  
      determining whether said next macro-instruction  
requires less than a second predetermined number of clock  
cycles; and

30       modifying one or more operations of said first  
macro-instruction if said next macro-instruction requires  
less than said second predetermined number of clock  
pulses. In this case said operations of said first  
macro-instruction which are modified are calculating an  
operand address for said next macro-instruction and  
fetching an operand from said memory. Alternatively said  
35       operations are modified to calculate an address of a  
program counter plus one and to fetch an instruction  
stored at said calculated address.



The second predetermined number is preferably three.

According to a second aspect of the present invention there is provided an apparatus for executing instructions in a controller having at least first and second function units which each include an arithmetic logic unit, comprising:

a macro-instruction register;

a next macro-instruction register coupled to said instruction register; and

first logic means, coupled to one of said instruction registers for generating control signals to concurrently execute, during a single clock cycle, first and second operations in said first and second function units, respectively, said first operation being used to carry out a first macro-instruction in said first macro-instruction register and said second operation being performed to carry out a next macro-instruction, and for using the result of said second operation during the execution of said next macro-instruction if said next macro-instruction has a number of operations less than a predetermined number and otherwise re-executing said second operation; and

second logic means for decoding said next macro-instruction to determine whether less than said predetermined number of operations are required.

Preferably one of said first and second operations is a memory access operation and said first logic means includes:

a first logic stage for providing access to a memory;

a second logic stage having a first arithmetic logic unit for performing arithmetic operations on data provided from at least one of said first and next macro-instructions and said memory; and

a third logic stage having a second arithmetic logic unit for performing arithmetic operations on data provided from at least one of said first and next macro-instructions and said memory, at least two of said





first, second and third logic stages being concurrently operated for a portion of said first and next macro-instructions.

Alternatively said first logic means includes a  
5 microcode memory having a plurality of groups of encoded micro-instructions, each group corresponding to a different macro-instruction, including a number of said groups for carrying out short macro-instructions which require less than said predetermined number of clock  
10 cycles, each said short macro-instruction group being written to omit said first operation pertaining to said short macro-instruction and all said groups of encoded micro-instructions being written to include said first operation pertaining to a subsequent macro-instruction.

15 Alternatively again said first logic means comprises:  
microcode address generation logic for producing microcode addresses responsive to a macro-instruction;  
a microcode memory coupled to said microcode memory;  
and

20 microcode control logic coupled to said microcode instruction register, for concurrently generating said control signals to said first and second function units.

The apparatus may additionally comprise: decoding means coupled to said next macro-instruction register for  
25 producing a signal if a next macro-instruction in said next macro-instruction register requires less than a second predetermined number of clock cycles; and means, responsive to said signal, for modifying at least one operation of a micro-instruction for a first  
30 macro-instruction in said first macro-instruction register. In this case the apparatus may further comprise a third function unit for performing memory accesses, all said function units being operable concurrently.

The operations can be classified into two groups.  
35 The first group is referred to as macro-sequencing operations and include operations that are performed for all macro-instructions or are performed for all macro-instructions of a certain type (i.e., all arithmetic



instructions requiring an operand). For example, these instructions include incrementing the program counter, fetching a next instruction from memory, calculating the address of an operand for the next instruction and  
5 fetching the operand for the next instruction. The second type of instruction which is executed in parallel covers micro operations which are dependent upon the particular macro-instruction. For example, this type of operation includes addition, subtraction and other arithmetic  
10 operations or specific movement of operands between registers. In a typical instruction, four operations of the first group are required, thus requiring four clock cycles. However, often only three operations of the second group are required, or in some instances, only  
15 two. Accordingly, by performing the first operation of the first group for a next instruction concurrently with the last operation of the first group for a current instruction, if the next instruction requires only three operations of the second group, the next operation can then be performed in three clock cycles. If the next  
20 instruction in fact requires four operations of the second group, the performance of its first operation of its first group by the previous instruction cycle is simply redundant. This procedure is followed because it is simpler to redo the operation when needed rather than store the result for the next clock cycle. The invention differs from the redundancy found in the prior art because the redundancy depends upon the number of operations specified by the next macro-instruction.

25  
30 For a fuller understanding of the nature and advantages of the invention, reference should be made to the ensuing detailed description taken in conjunction with the accompanying drawings.

35 BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram of a typical prior art controller;



Figure 2 is a block diagram of a controller embodying the second aspect of the present invention; and

Figure 3 is a block diagram of the controller of Figure 2 with a specific function unit arrangement for the logic functions.

#### DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Figure 1 is a block diagram of a typical prior art processor. An instruction register 10 receives an instruction from memory which is decoded by microcode address generation logic 12 to produce addresses to a microcode memory 14. The instructions stored in microcode memory 14 are supplied to a micro-instruction register 16, which may itself affect the address of the next instruction in microcode memory 14. The instruction in micro-instruction register 16 is decoded by microcode decode logic 18. The decoded logic generates control signals which are supplied to function units 20 and 22. The function units perform the memory fetches, arithmetic operations, and other manipulations specified by the micro-instruction.

Because the processor of Figure 1 has two function units, during any one clock cycle two operations may be performed concurrently. A typical macro-instruction sequence may require a number of macro-sequencing operations as well as micro-operations which may be performed concurrently. The macro-sequencing operations would be operations which are done for all macro-instructions or for certain types of macro-instructions (i.e., arithmetic operations requiring the fetching of an operand), while other micro-instructions are dependent upon a particular macro-instruction (i.e., particular arithmetic operations). Thus, for example, function unit 20 of Figure 1 could be performing a macro-sequencing operation (such as incrementing a program counter) while function unit 22 could be performing another operation, such as adding two operands. This parallel operation is shown in



the following table for macro-instructions 0 and 1 where A, B, C and D represent macro-sequencing micro-instructions and w, x, y and z represent instruction-dependent micro operations.

TABLE 1

	<u>Macro-Instr.</u>	<u>Clock Cyl.</u>	<u>Operation</u>
	0	0	A(0) w(0)
10	0	1	B(0) x(0)
	0	2	C(0) y(0)
	0	3	D(0) z(0)
	1	0	A(1) x(1)
	1	1	B(1) y(1)
15	1	2	C(1) z(1)
	1	3	D(1)

As can be seen from the above table, by doing operations in parallel, only four clock instructions are required for each macro-instruction. In the example shown, for macro-instruction 1, four macro-level operations are required while only three instruction-dependent micro-operations are required. The preferred embodiment takes advantage of this feature of certain macro-instructions by adding a third function unit so that three operations can be performed in parallel.

The present invention concerns the parallel performance of macro-sequencing operations. For most instruction-dependent micro-operations (w, x, y, z) the operations must be performed sequentially, and thus a time savings by parallel operation is not possible. However, oftentimes more than one macro-sequencing operation can be performed simultaneously. Accordingly, the embodiment uses a third function unit to perform a first macro-sequencing operation of a next instruction concurrently with the last macro-sequencing operation of a first instruction. As can be seen in Table 2 below, this



results in macro-instruction 1 of Table 1 requiring only three clock cycles to execute.

TABLE 2

<u>Macro-Instr.</u>	<u>Clock Cyl.</u>	<u>Operation</u>	
0	0	A(0)	w(0)
0	1	B(0)	x(0)
0	2	C(0)	y(0)
0	3	D(0) A(1)	z(0)
1	0	B(1)	x(1)
1	1	C(1)	y(1)
1	2	D(1) A(2)	z(1)
2	0	A(2)	w(2)
2	1	B(2)	x(2)
2	2	C(2)	y(2)
2	3	D(2) A(3)	z(2)

As can be seen from Table 2, if a macro-instruction 2 requires four instruction-dependent micro operations (w, x, y, z), then step A(2) performed during macro-instruction 1 simply becomes redundant and is repeated. Alternatively, step A(2) could simply be omitted from macro-instruction 2 with no macro-sequencing operation being performed during the first clock cycle, L-3.



5 Referring to Fig. 2, both an instruction register 24  
and a next instruction register 26 are provided to give  
a macro-instruction pipeline. These registers are  
coupled to microcode address generation logic 28,  
microcode memory 30 and micro-instruction register 32  
10 in similar manner to the circuit of Fig. 1. However,  
rather than using simply microcode decode logic which  
only decodes micro-instruction register 32, a decode  
logic block 34 is used which takes inputs from the  
macro-instruction register as well as the micro-  
15 instruction register. Decode logic 34 looks at the  
contents of the next instruction register (NIR) 26 and,  
if it requires less than a certain number of clock  
cycles, modifies the code from micro-instruction  
register 32 to alter the control signals provided to  
20 function units 36 and 38 - 40 (F1, F2 through FN).  
Decode logic 34 looks at the next instruction in next  
instruction register 26 and determines how many clock  
cycles it could be done in. For instance, if a next  
instruction requires only instruction-dependent op-  
25 erations y and z, then it can be done in two clock  
cycles if macro-sequencing instruction A and B are  
performed during the current instruction. This can be  
done by doing A and B in parallel with C and D of the  
current instruction or, if C and D are not needed  
30 because of the nature of the next instruction, op-  
erations C and D can be modified to become operations A  
and B.

An example is where A and B relate to incre-  
menting the program counter and C and D relate to  
35 calculating an operand address and fetching the operand  
for the next instruction. Where the next instruction  
has only two instruction-dependent operations (y, z),



and does not require an operand, steps C and D being performed by the current instruction are unnecessary. Accordingly, steps C and D can be modified to become steps A and B for the next instruction. When the next  
 5 instruction is executed, it can thus do its steps C and D concurrently with steps y and z.

Table 3 below shows the resulting sequence of operations where a current instruction 0 in instruction register 24 requires four clock cycles while a next  
 10 instruction in NIR register 26 requires only two clock cycles. Decode logic 34 of Fig. 2 looks at the contents of NIR 26 and determines that only two clock cycles are required. Accordingly, it modifies operations C(0) and D(0) to become operations A(1) and  
 15 B(1), respectively. Thus, when macro-instruction 1 is itself executed, since steps A and B have already been performed, it can perform steps C(1) and D(1) concurrently with instruction-dependent steps y(1) and z(1), thus enabling the instruction to be completed in only  
 20 two clock cycles.

TABLE 3

	<u>Macro Instr.</u>	<u>Clock Cyc.</u>	<u>Operation</u>	
25	0	0	A(0)	w(0)
	0	1	B(0)	x(0)
	0	2	[C(0)] A(1)	y(0)
	0	3	[D(0)] B(1)	z(0)
30	1	0	C(1)	y(1)
	1	1	D(1)	z(1)
	2	0	A(2)	w(2)
	2	1	B(2)	x(2)
35	2	2	C(2)	y(2)
	2	3	D(2)	z(2)

Fig. 3 shows a specific embodiment of the circuit of Fig. 2. In Fig. 3, the contents of next



instruction register 26 are provided via a bus 42 to microcode address generation logic 28. A data bus 59 couples IR 24 to the function units to provide data to be operated on when appropriate. The decode logic 34 consists of a NIR decode circuit 44, microcode decode logic 46 and combination logic 48. NIR decode logic 44 determines whether a two cycle instruction is present in NIR 26 and, if so, presents a signal on a line 50 to decode logic 48. Decode logic 48 passes control signals 52 from microcode decode logic 46 if no signal is present on line 50. Otherwise, the signal on line 50 modifies the digital content of the control signals.

Three function units 54, 56 and 58 are utilized. Function units 54 and 56 contain arithmetic logic units 60 and 62, respectively. In addition, each contains a register file 64 or 66, respectively. Register file 66 includes the program counter. Function unit 58 is used to access memory 68.

The sequencing of instructions through the three function units is shown in Table 4 below.

TABLE 4

Clock Cyc.	F1	F2	F3
0	w(0)	A(0)	-
1	x(0)	-	B(0)
2	y(0)	C(0) or A(1)	-
3	z(0)	A(1)	D(0) or B(1)

Table 4 shows a four clock cycle sequence which combines the redundancy of Table 2 and the macro-sequencing instruction modification of Table 3. In clock cycles L-1 and L, if the next instruction is a two cycle instruction, steps C(0) and D(0) are modified to become A(1) and B(1). During clock cycle L, while either step D(0) or B(1) is being performed in function unit F3, step A(1) is being performed in function unit F2. As can be seen, the operation in function unit F2





during clock cycle L will be redundant when the next instruction is a two clock instruction, and will be used only if the next instruction is a three-clock instruction. In addition, it can be seen that function unit F2 is not used during clock cycle L-2 and function unit F3 is not used during clock cycles L-3 and L-1. Accordingly, this gives added flexibility to the programming to enable instruction-dependent operations w, x and y to use two function units concurrently if necessary. . . .

The actual operations A, B, C and D performed in the preferred embodiment and the modifications performed for a two tick (clock) cycle are set forth below.

15           A(0):       Calculate the address of the macro program counter (P) plus 1. This is equal to the address of the present instruction plus 2.

20           B(0):       Fetch the instruction from memory whose address was calculated in A. Store the address calculated in A to P.

              C(0):       If not two tick (NIR) then:  
                          Calculate the address (base + displacement) of the operand for the next instruction.

25                       (Else  
                          A(1): calculate the address of the macro program counter (P) plus 1. This is equal to the address of the present instruction plus 3.)

30           Load the current instruction register (IR) with the instruction in the next instruction register (NIR) and the instruction fetched in B into NIR.

              D(0):       If not two tick (NIR) then:  
                          Fetch the operand for the next instruction, now in IR.

35                       (Else  
                          B(1): Fetch the instruction from memory whose address was calculated in C.  
                          Store the address calculated in C to P.)



A(1): Calculate the address of the macro program counter (P) plus 1. This is equal to the address of the instruction 3 after the present one.

5 A(1) is the redundant operation which is done in parallel with D(0) for the sake of a three micro cycle macro instruction which might follow the present instruction. "Two tick (NIR)" is a decode of the next instruction register that indicates that the next instruction will be executed in two micro cycles. "Two tick (IR)" of D(0) reflects the movement of the next instruction into the instruction register during C(0).  
10 Because of this shift, NIR decode logic 44 includes a register for storing the portion of the next instruction needed for D(0). The above description does not include the instruction-dependent operations (x(0), y(0), etc.) that occur in parallel with the macro sequencing operations.

In a preferred embodiment, the macro-instructions have lengths of either two, three or four or more clock cycles. This results in six possible micro-instruction flows: four or more clock instructions, three clock instructions and two clock instructions which are followed by either two or more clock instructions or a two clock instruction. This instruction flow is as set forth in the following table.

30

35



TABLE 5

		Inst. 1 is >		Inst. 1 is =	
		<u>2 Clocks</u>		<u>2 Clocks</u>	
5	<u>≥ 4 Clock</u>	<u>Clock</u>	<u>Operation</u>	<u>Operation</u>	
	<u>Instruction</u>	<u>Cyc.</u>			
		0,L-3	A(0), w(0)	A(0),	w(0)
		0,L-2	B(0), x(0)	B(0),	x(0)
		0,L-1	C(0), y(0)	A(1),	y(0)
	0,L	D(0), A(1), z(0)	B(1), A(1),	z(0)	
10	<u>3 Clock</u>				
	<u>Instruction</u>	<u>-1,L</u>	<u>D(-1,A(0),z(-1))</u>	<u>D(-1,A(0),z(-1))</u>	
		0,0	B(0), x(0)	B(0),	x(0)
		0,1	C(0), y(0)	A(1),	y(0)
		0,2	D(0), A(1), z(0)	B(1), A(1),	z(0)
15	<u>2 Clock</u>	<u>-1,L-1</u>	<u>A(0), y(-1)</u>	<u>A(0), y(-1)</u>	
	<u>Instruction</u>	<u>-1,L</u>	<u>B(0),A(0) z(-1)</u>	<u>B(0),A(0),z(-1)</u>	
		0,0	C(0), y(0)	A(1),	y(0)
		0,1	D(0), A(1), z(0)	B(1), A(1),	z(0)

As will be understood by those familiar with the art, the present invention may be embodied in other specific forms without departing from the spirit or essential characteristics thereof. For example, a next instruction which would require modification of the macro-sequencing operations of a current instruction could be other than a two clock cycle instruction. Accordingly, the disclosure of the preferred embodiments of the invention is intended to be illustrative, but not limiting, of the scope of the invention which is set forth in the following claims.



THE CLAIMS DEFINING THE INVENTION ARE AS FOLLOWS:

1. A method for executing instructions in a controller comprising the steps of:

fetching with said controller a first

5 macro-instruction and a next macro-instruction;

executing with said controller concurrently during a single clock cycle first and second operations, said first operation being used to carry out said first macro-instruction and said second operation being

10 performed to carry out said next macro-instruction;

decoding with said controller said next macro-instruction to determine whether less than a predetermined number of operations are required; and

using in said controller the result of said second operation during execution of said next macro-instruction if said next macro-instruction has a number of operations less than said predetermined number and otherwise re-executing said second operation.

2. The method of claim 1 wherein only one of said first and second operations is a memory access operation.

3. The method of claim 1 wherein said first operation is incrementing a program counter and said second operation is a data operand fetch.

4. The method of any preceding claim comprising the further steps of:

determining whether said next macro-instruction requires less than a second predetermined number of clock cycles; and

modifying one or more operations of said first macro-instruction if said next macro-instruction requires less than said second predetermined number of clock pulses.

5. The method of claim 4 wherein said operations of said first macro-instruction which are modified are calculating an operand address for said next macro-instruction and fetching an operand from said memory.



6. The method of claim 5 wherein said operations are modified to calculate an address of a program counter plus one and to fetch an instruction stored at said calculated address.

5 7. The method of claim 4 wherein said second predetermined number is three.

8. An apparatus for executing instructions in a controller having at least first and second function units which each include an arithmetic logic unit, comprising:

10 a macro-instruction register;

a next macro-instruction register coupled to said instruction register; and

first logic means, coupled to one of said instruction registers for generating control signals to concurrently execute, during a single clock cycle, first and second operations in said first and second function units, respectively, said first operation being used to carry out a first macro-instruction in said first macro-instruction register and said second operation being performed to carry out a next macro-instruction, and for using the result of said second operation during the execution of said next macro-instruction if said next macro-instruction has a number of operations less than a predetermined number and otherwise re-executing said second operation; and

second logic means for decoding said next macro-instruction to determine whether less than said predetermined number of operations are required.

9. The apparatus of claim 8 wherein one of said first and second operations is a memory access operation and said first logic means includes:

a first logic stage for providing access to a memory;

a second logic stage having a first arithmetic logic unit for performing arithmetic operations on data provided from at least one of said first and next macro-instructions and said memory; and



a third logic stage having a second arithmetic logic unit for performing arithmetic operations on data provided from at least one of said first and next macro-instructions and said memory, at least two of said first, second and third logic stages being concurrently operated for a portion of said first and next macro-instructions.

10. The apparatus of claim 8 wherein said first logic means includes a microcode memory having a plurality of groups of encoded micro-instructions, each group corresponding to a different macro-instruction, including a number of said groups for carrying out short macro-instructions which require less than said predetermined number of clock cycles, each said short macro-instruction group being written to omit said first operation pertaining to said short macro-instruction and all said groups of encoded micro-instructions being written to include said first operation pertaining to a subsequent macro-instruction.

11. The apparatus of claim 8 wherein said first logic means comprises:

microcode address generation logic for producing microcode addresses responsive to a macro-instruction;

a microcode memory coupled to said microcode memory; and

microcode control logic coupled to said microcode instruction register, for concurrently generating said control signals to said first and second function units.

12. The apparatus of claim 8 further comprising: decoding means coupled to said next macro-instruction register for producing a signal if a next macro-instruction in said next macro-instruction register requires less than a second predetermined number of clock cycles; and

means, responsive to said signal, for modifying at least one operation of a micro-instruction for a first macro-instruction in said first macro-instruction register.



13. The apparatus of claim 12 further comprising a third function unit for performing memory accesses, all said function units being operable concurrently.

14. A method for executing instructions in a controller substantially as herein described with reference to tables 2 and 4 of the accompanying drawings.

15. Apparatus for executing instructions in a controller substantially as herein described with reference to Figures 2 and 3 of the accompanying drawings.

DATED this 5th day of December 1990

TANDEM COMPUTERS INCORPORATED

By their Patent Attorneys  
GRIFFITH HACK & CO.



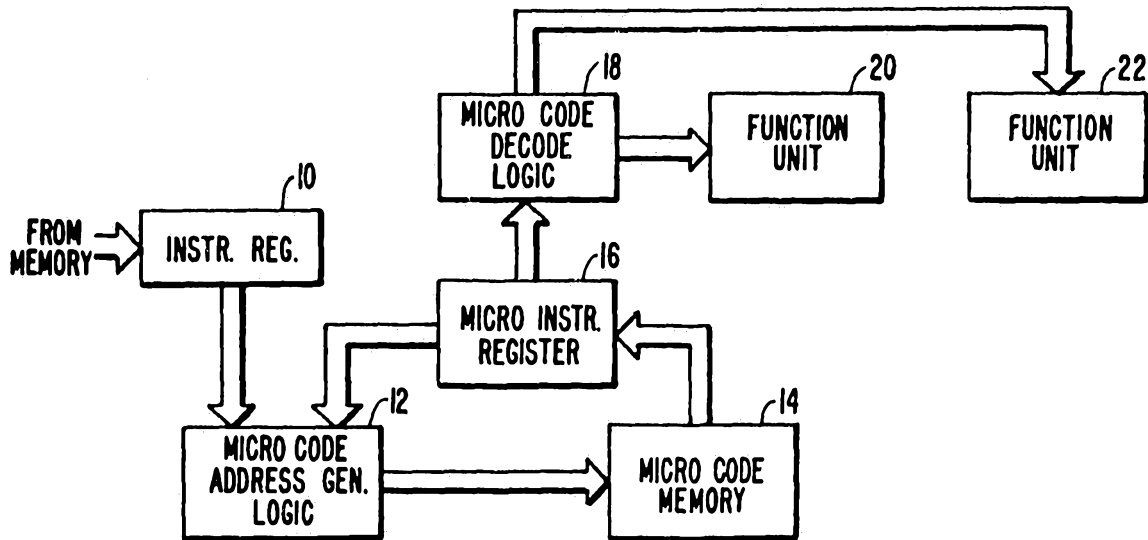


FIG. 1. (PRIOR ART)

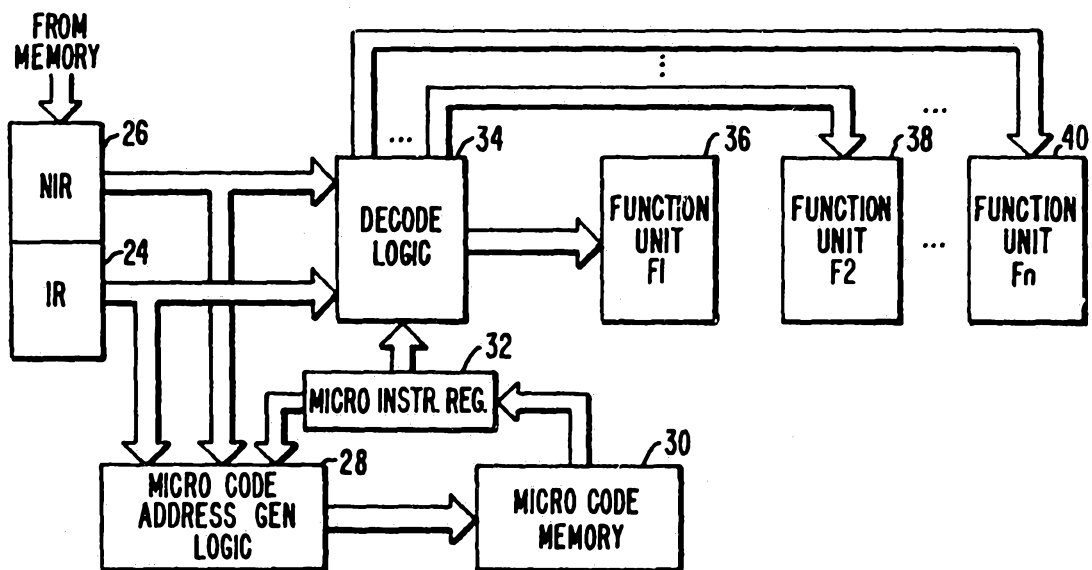


FIG. 2.



