Digital Computer system

The digital computer is a digital system that performs various computational tasks. The word digital implies that the information in the computer is represented by variables that take a limited number of discrete values. These values are processed internally by components that can maintain a limited number of discrete states. The decimal digits 0, 1, 2,..., 9, for example, provide 10 discrete values. The first electronic digital computers, developed in the late 1940s, were used primarily for numerical computations. In this case the discrete elements are the digits. From this application the term digital computer has emerged. In practice, digital computers function more reliably if only two states are used. Because of the physical restriction of components, and because human logic tends to be binary (i.e., true-or-false, yes-or-no statements), digital components that are constrained to take discrete values are further constrained to take only two values and are said to be binary.

Digital computers use the binary number system, which has two digits: 0 and 1. A binary digit is called a bit. Information is represented in digital computers in groups of bits. By using various coding techniques, groups of bits can be made to represent not only binary numbers but also other discrete symbols, such as decimal digits or letters of the alphabet. By judicious use of binary arrangements and by using various coding techniques, the groups of bits are used to develop complete sets of instructions for performing various types of computations.

In contrast to the common decimal numbers that employ the base 10 system, binary numbers use a base 2 system with two digits: 0 and 1. The decimal equivalent of a binary number can be found by expanding it into a power series with a base of 2. For example, the binary number 1001011 represents a quantity that can be converted to a decimal number by multiplying each bit by the base 2 raised to an integer power as follows:

$$1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 75$$

The seven bits 1001011 represent a binary number whose decimal equivalent is 75.

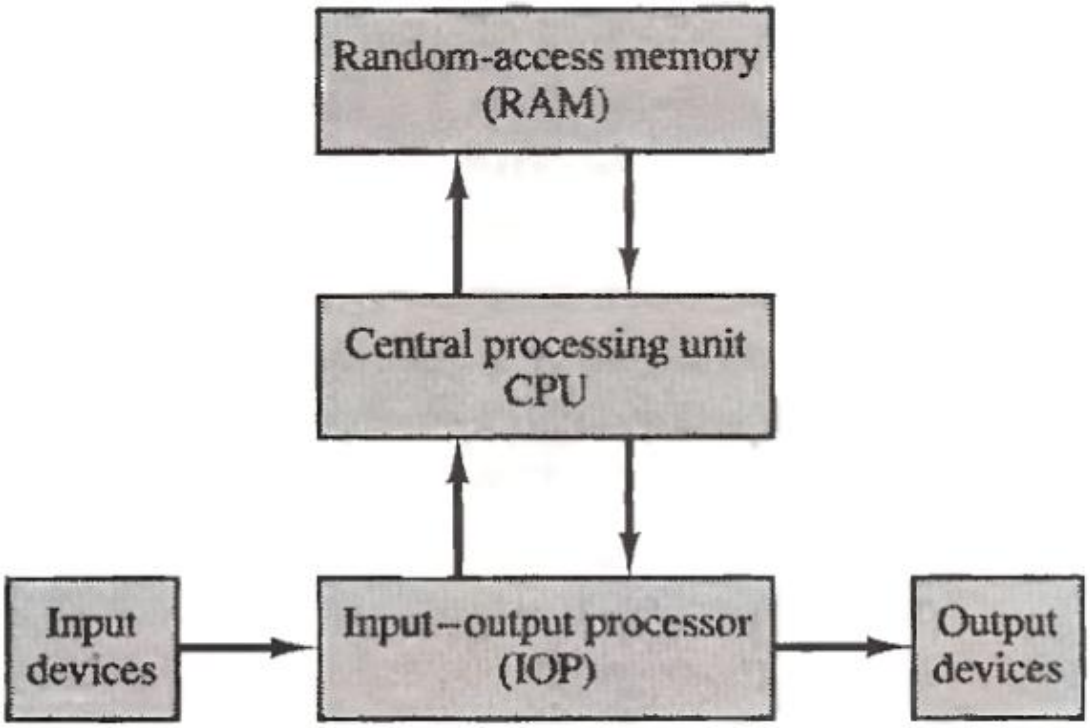
Computer program

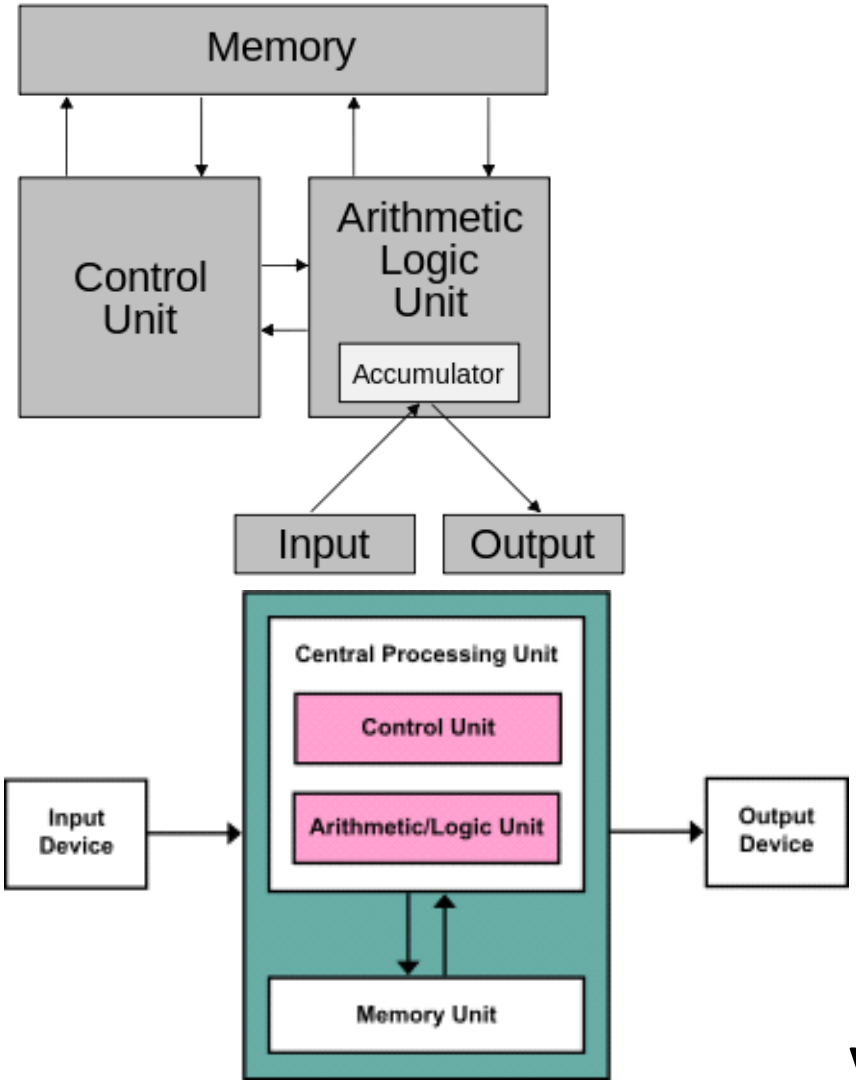
A computer system is sometimes subdivided into two functional entities: hardware and software. The hardware of the computer consists of all the electronic components and electromechanical devices that comprise the physical entity of the device. Computer software consists of the instructions and data that the computer manipulates to perform various data-processing tasks.

A program is a sequence of instructions for the computer is called a program. The data that are manipulated by the program constitute the data base.

A computer system is composed of its hardware and the system software available for its use. The system software of a computer consists of a collection of programs whose purpose is to make more effective use of the computer. The programs included in a systems software package are referred to as the operating system. They are distinguished from application programs written by the user for the purpose of solving particular problems. For example, a high-level language program written by a user to solve particular data-processing needs is an application program, but the compiler that translates the high-level language program to machine language is a system program. The customer who buys a computer system would need, in addition to the hardware, any available software needed for effective operation of the computer. The system software is an indispensable part of a total computer system. Its function is to compensate for the differences that exist between user needs and the capability of the hardware.

The hardware of the computer is usually divided into three major parts ,as shown in Fig. 1-1. The central processing unit (CPU) contains an arithmetic logic unit for manipulating data, a number of registers for storing data, and control circuits for fetching and executing instructions.





Von Neumann architecture

Digital components of computer

Integrated Circuit (IC)

Digital circuits are constructed with integrated circuits. An integrated circuit (abbreviated IC) is a small silicon semiconductor crystal, called a chip, containing the electronic components for the digital gates. The various gates are interconnected inside the chip to form the required circuit. The chip is mounted in a ceramic or plastic container, and connections are welded by thin gold wires to external pins to form the integrated circuit.

As the technology of ICs has improved, the number of gates that can be put in a single chip has increased considerably. The differentiation between those chips that have a few internal gates and those having hundreds or thousands of gates is made by a customary reference to a package as being either a small-, medium-, or large-scale integration device.

Small-scale integration (SSI) - The number of gates is usually less than 10.
Medium-scale integration (MSI) - devices have a complexity of approximately 10 to 200 gates in a single package.
Large-scale integration (LSI) devices contain between 200 and a few thousand gates in a single package.
Very-large-scale integration (VLSI) devices contain thousands of gates within a single package. Examples are large memory arrays and complex microcomputer chips.

Decoder
A decoder is a combinational circuit that converts binary information from the n coded inputs to a maximum of 2" unique outputs. If the n-bit coded information has unused bit combinations, the decoder may have less than 2" outputs.

Multiplexer
A multiplexer is a combinational circuit that receives binary information from one of 2" input data lines and directs it to a single output line. The selection of a particular input data line for the output is determined by a set of selection inputs. A 2"-to-l multiplexer has 2" input data lines and n input selection lines whose bit combinations determine which input data are selected for the

Register
A register is a group of flip-flops with each flip-flop capable of storing one bit of information. An n-bit register has a group of n flip-flops and is capable of storing any binary information of n bits. In addition to the flip-flops, a register may have combinational gates that perform certain data-processing tasks. In

its broadest definition, a register consists of a group of flip-flops and gates that effect their transition. The flip-flops hold the binary information and the gates control when and how new information is transferred into the register.

Shift Registers

A register capable of shifting its binary information in one or both directions is called a shift register. The logical configuration of a shift register consists of a chain of flip-flops in cascade, with the output of one flip-flop connected to the input of the next flip-flop. All flip-flops receive common clock pulses that initiate the shift from one stage to the next.

Binary Counters

A register that goes through a predetermined sequence of states upon the application of input pulses is called a counter. The input pulses may be clock pulses or may originate from an external source. They may occur at uniform intervals of time or at random. Counters are found in almost all equipment containing digital logic. They are used for counting the number of occurrences of an event and are useful for generating riming signals to control the sequence of operations in digital computers.

Memory Unit

A memory unit is a collection of storage cells together with associated circuits needed to transfer information in and out of storage. The memory stores binary information in groups of bits called words. A word in memory is an entity of bits that move in and out of storage as a unit. A memory word is a group of l's and O's and may represent a number, an instruction code, one or more alphanumeric characters, or any other binary-coded information. A group of eight bits is called a byte. Most computer memories use words whose number of bits is a multiple of 8. Thus a 16-bit word contains two bytes, and a 32-bit word is made up of four bytes. Two major types of memories are used in computer systems: random-access memory (RAM) and read-only memory (ROM).

Microprogrammed Control

الدقيق بالبرنامج التحكم

The function of the control unit in a digital computer is to initiate sequences of microoperations. The number of different types of microoperations that are available in a given system is finite. The complexity of the digital system is derived from the number of sequences of microoperations that are performed.

الأنواع عدد .الدقيقة العمليات تسلسل في الشروع هو رقمي كمبيوتر جهاز في التحكم وحدة وظيفة من عدد من الرقمي النظام تعقيد يشتق .محدود هو معين نظام في تتوفر التي الدقيقة العمليات من المختلفة تنفيذها يتم التي الدقيقة العمليات من متواليات.

When the control signals are generated by hardware using conventional logic design techniques, the control unit is said to be hardwired.

بإن يقال المنطق، للتصميم التقليدية التقنيات باستخدام الأجهزة قبل من التحكم إشارات إنشاء يتم عندما الاجهزة باستخدام منجزة التحكم وحدة (الاسلاك او)

The principle of microprogramming is an elegant and systematic method for controlling the microoperation sequences in a digital computer.

جهاز في الدقيقة العملية تسلسل على للسيطرة ومنظم أنيق أسلوب هو الدقيق بالبرنامج التحكم مبدأ رقمي كمبيوتر

The control variables at any given time can be represented by a control word string of l's and O's called a control word. A control unit whose binary control variables are stored in memory is called a microprogrammed control unit and each word in control memory contains within it a microinstruction. A sequence of microinstructions constitutes a microprgram.

الواحدات من المتكونة التحكم كلمة بسلسلة ممثلة تكون أن يمكن معين وقت أي في السيطرة متغيرات تسمى ذاكرة في تخزينها يتم فيها الثنائي التحكم متغيرات التي التحكم وحدة .التحكم كلمة تسمى والاصفار .الدقيق الايعاز على بداخلها تحتوي التحكم ذاكرة في كلمة وكل الدقيق البرنامج ذات التحكم وحدة ان الدقيق البرنامج تالف الدقيقة الايعازات من سلسلة.

A computer that employs a microprogrammed control unit will have two separate memories: 1.main memory , 2. control memory. The main memory is available to the user for storing the programs and the contents of main

memory may alter when the data are manipulated and every time that the program is changed.

1. :المنفصلة الذاكرات من اثنين سيحتوي دقيق ببرنامج التحكم وحدة يوضف الذي الكمبيوتر جهاز ان الذاكرة ومحتويات البرامج لتخزين للمستخدم متاح الرئيسية الذاكرة .السيطرة ذاكرة .2 الرئيسية، الذاكرة .البرنامج تغيير يتم مرة كل وفي البيانات في التلاعب يتم تتغير عندما قد الرئيسية

In contrast, the control memory holds a fixed microprogram that cannot be altered by the occasional user. The microprogram consists of microinstructions that specify various internal control signals for execution of register microoperations.

.العرضي المستخدم طريق عن تغييره يمكن لا ثابت دقيق برنامج تحمل السيطرة الذاكرة فإن المقابل، في العمليات لتنفيذ الداخلية الرقابة الإشارات مختلف تحدد التي الدقيقة الايعازات من الدقيق البرنامج ويتكون .للمسجل الدقيقة

Figure 1 Microprogrammed control organization.

The general configuration of a microprogrammed control unit is demonstrated in the block diagram of Fig. 1

The control memory is assumed to be a ROM, within which all control information is permanently stored. The control memory address register specifies the address of the microinstruction, and the control data register holds the microinstruction read from memory. The microinstruction contains a control word that specifies one or more microoperations for the data processor. Once these operations are executed, the control must determine the next address.

السيطرة معلومات جميع تخزين يتم بضمنها والتي فقط قراءة ذاكرة تكون التحكم الذاكرة أن ويفترض قراءة يحمل التحكم بيانات وسجل, الايعازالدقيق عنوان السيطرة لذاكرة العنوان مسجل يحدد .دائم بشكل العمليات من أكثر أو واحدة تحدد التي التحكم كلمة على يحتوي الدقيق الايعاز .الذاكرة من الدقيق الايعاز ..الايعازالمقبل عنوان تحديد وحدةالتحكم على يجب العمليات، هذه تنفيذ يتم عندما .البيانات لمعالج الدقيقة

While the microoperations are being executed, the next address is computed in the next address generator circuit (some times called microprogram

sequencer) and then transferred into the control address register to read the next microinstruction.

احيانا وتسمى) القادم الايعاز توليد دائرة في التالي العنوان احتساب يتم الدقيقة، العمليات تنفيذ يجري بينما المقبل الدقيق الايعاز لقراءة التحكم عنوان مسجل في نقلت ثم ومن (الدقيق البرنامج منظم..

The main advantage of the microprogrammed control is the fact that once the hardware configuration is established, there should be no need for further hardware or wiring changes. If we want to establish a different control sequence for the system, all we need to do is specify a different set of microinstructions for control memory. The hardware configuration should not be changed for different operations; the only thing that must be changed is the microprogram residing in control memory.

لا أن يجب الأجهزة، تكوين تأسيس يتم أن بمجرد أنه حقيقة هو الدقيقة بالبرمجة للسيطرة الرئيسية الميزة عن مختلف تحكم تسلسل إنشاء نريد كنا إذا .تغييرها يتم الأسلاك أو الأجهزة من لمزيد حاجة هناك تكون لا .التحكم لذاكرة الدقيقة الايعازات من مختلفة مجموعة تحديد هو به القيام علينا ما كل الحالي، النظام المقيم الدقيق البرنامج هو تغييره يجب الذي الوحيد الشيء مختلفة؛ لعمليات الأجهزة تكوين تغيير ينبغي السيطرة ذاكرة في.

The transformation from the instruction code bits to an address in control memory where the routine is located is referred to as a ***mapping process***. A mapping procedure is a rule that transforms the instruction code into a control memory address.

اليه يشار , موجود (البرنامج)الروتين حيث السيطرة ذاكرة في عنوان إلى الايعاز كود بتات من التحويل ذاكرة عنوان إلى الايعاز رمز تحول التي القاعدة هو الخرائط رسم إجراء .الخرائط رسم عملية أنها على سيطرة.

In summary, the address sequencing capabilities required in a control memory are:

1. Incrementing of the control address register.
2. Unconditional branch or conditional branch, depending on status bit conditions.
3. A mapping process from the bits of the instruction to an address for control memory.
4. A facility for subroutine call and return.

هي السيطرة الذاكرة في المطلوبة التسلسل عنوان قدرات فإن وباختصار،:

1. واحد بمقدار التحكم عنوان سجل زيادة.
2. الحالة بت لوضع تبعا المشروط، تفرع أو مشروط غير تفرع.

3. السيطرة لذاكرة عنوان إلى الايعاز بتات من الخرائط رسم عملية.
4. منه والعودة الفرعي البرنامج لدعوة مرفق.

Figure 2 Selection of address for control memory

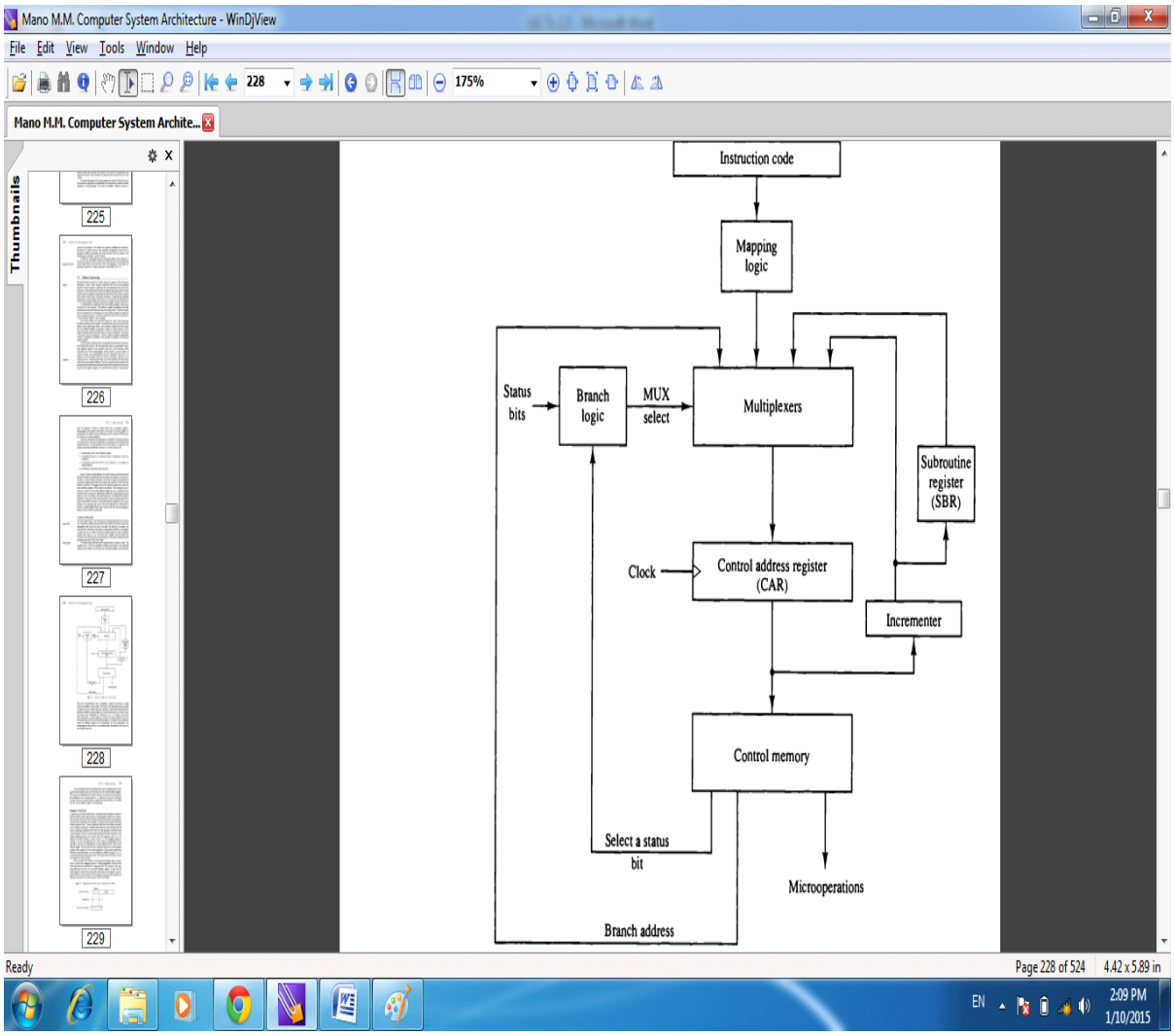


Figure 2 shows a block diagram of a control memory and the associated hardware needed for selecting the next microinstruction address.

The diagram shows four different paths from which the control address register (CAR) receives the address. The incrementer increments the content of the control address register by one, to select the next microinstruction in sequence. Branching is achieved by specifying the branch address in one of the fields of the microinstruction. Conditional branching is obtained by using part of the microinstruction to select a specific status bit in order to determine its condition. An external address is transferred into control memory via a mapping logic circuit. The return address for a subroutine is stored in a special register whose value is then used when the microprogram wishes to return from the subroutine.

Branching

There are two types of branching: conditional and unconditional.

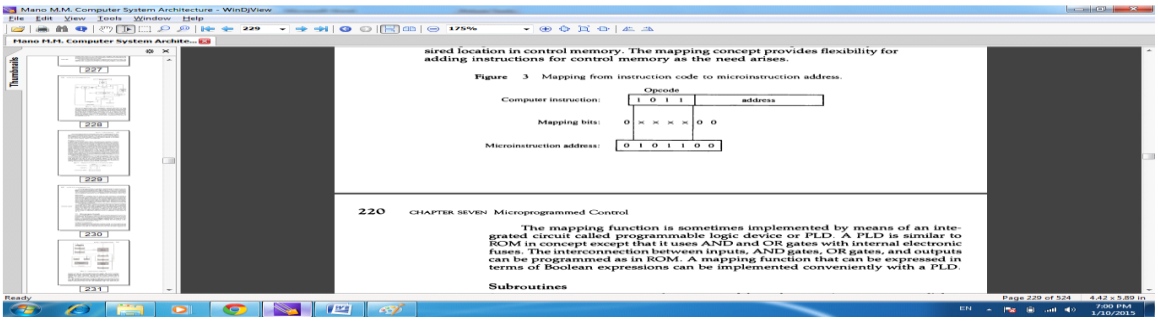
Conditional branching:

The status conditions are special bits in the system that provide parameter information such as the carry-out of an adder, the sign bit of a number, the mode bits of an instruction, and input or output status conditions. Information in these bits can be tested and actions initiated based on their condition: whether their value is 1 or 0. The status bits, together with the field in the microinstruction that specifies a branch address, control the conditional branch decisions generated in the branch logic.

Unconditional branching:

An unconditional branch microinstruction can be implemented by loading the branch address from control memory into the control address register. This can be accomplished by fixing the value of one status bit at the input of the multiplexer, so it is always equal to 1.

Mapping of Instruction



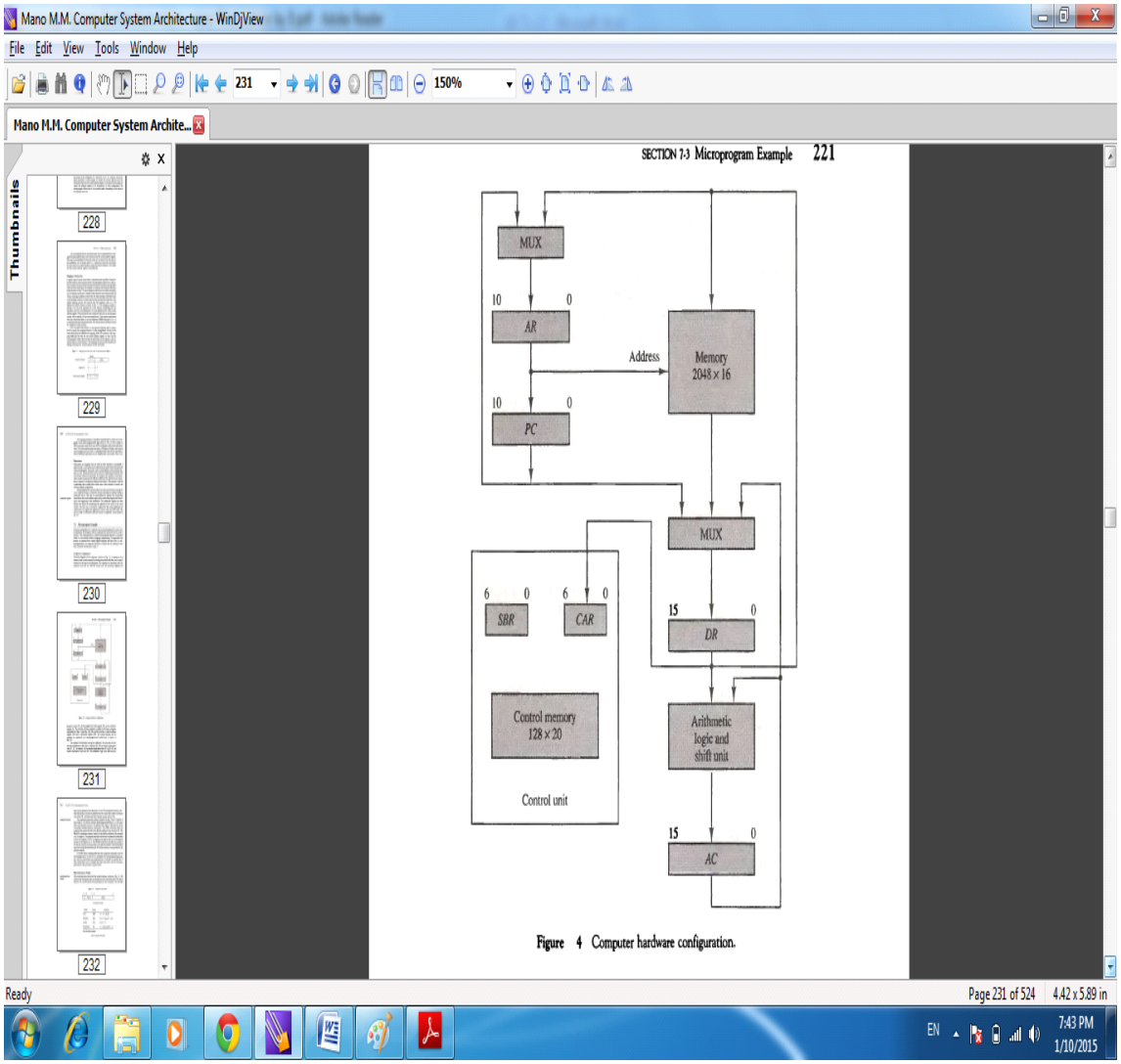
Assume a computer with a simple instruction format has an operation code of four bits which can specify up to 16 distinct instructions. Assume further that the control memory has 128 words, requiring an address of seven bits. For each operation code there exists a microprogram routine in control memory that executes the instruction. One simple mapping process that converts the 4-bit operation code to a 7-bit address for control memory is shown in Fig. 7-3. This mapping consists of placing a 0 in the most significant bit of the address, transferring the four operation code bits, and clearing the two least significant bits of the control address register.

Subroutines

Subroutines are programs that are used by other routines to accomplish a particular task. A subroutine can be called from any point within the main body of the microprogram.

For example, the sequence of microoperations needed to generate the effective address of the operand for an instruction is common to all memory reference instructions. This sequence could be a subroutine that is called from within many other routines to execute the effective address computation.

Microprograms that use subroutines must have a provision احتياط for storing the return address during a subroutine call and restoring the address during a subroutine return. This may be accomplished by placing the incrementedoutput from the control address register into a subroutine register and branching to the beginning of the subroutine. The subroutine register can then become the source for transferring the address for the return to the main routine.



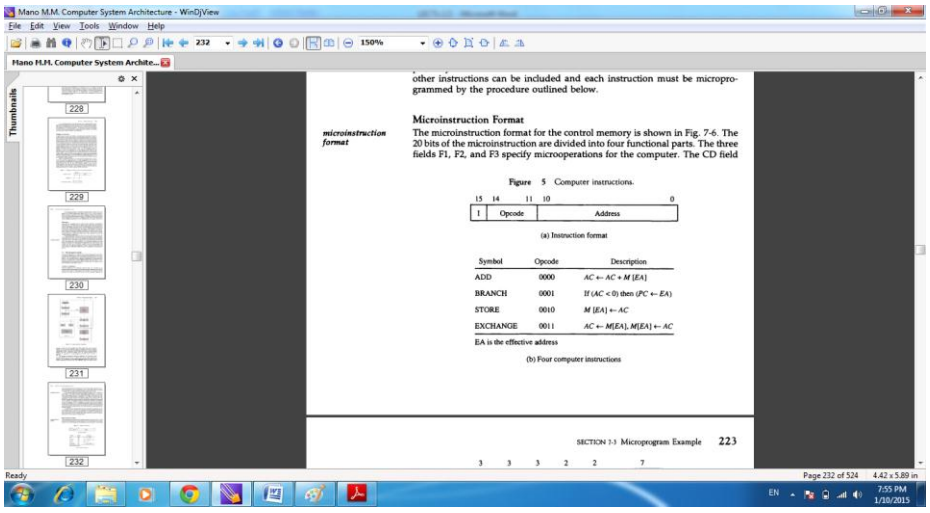
Computer microprogram Example

The block diagram of the computer shown consists of two memory units: a main memory for storing instructions and data, and a control memory for storing the microprogram. Four registers are associated with the processor unit and two with the control unit. The processor registers are program counter PC, address register AR, data register DR, and accumulator register AC.

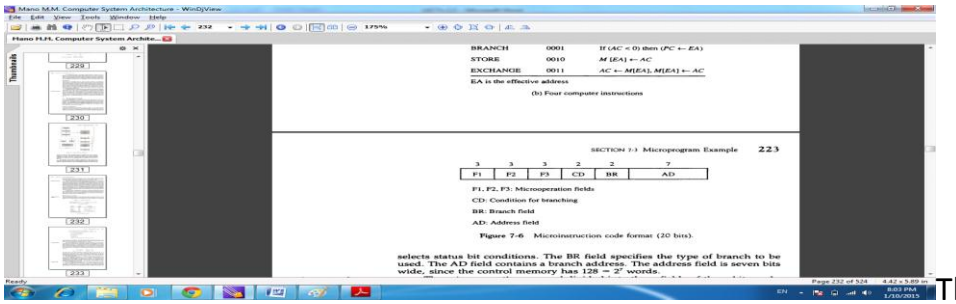
The control unit has a control address register CAR and a subroutine register SBR.

Computer instruction format

The computer instruction format is shown in fig 5(b), consists of three fields: a 1-bit held for indirect addressing symbolized by (I), a 4-bit operation code (opcode), and an 11-bit address field. Figure 5(b) lists four of the 16 possible memory-reference instructions. The ADD instruction adds the content of the operand found in the effective address to the content of AC. The BRANCH instruction causes a branch to the effective address if the operand in AC is negative. The program proceeds with the next consecutive instruction if AC is not negative. The AC is negative if its sign bit (the bit in the leftmost position of the register) is a 1. The STORE instruction transfers the content of AC into the memory word specified by the effective address. The EXCHANGE instruction swaps the data between AC and the memory word specified by the effective address.

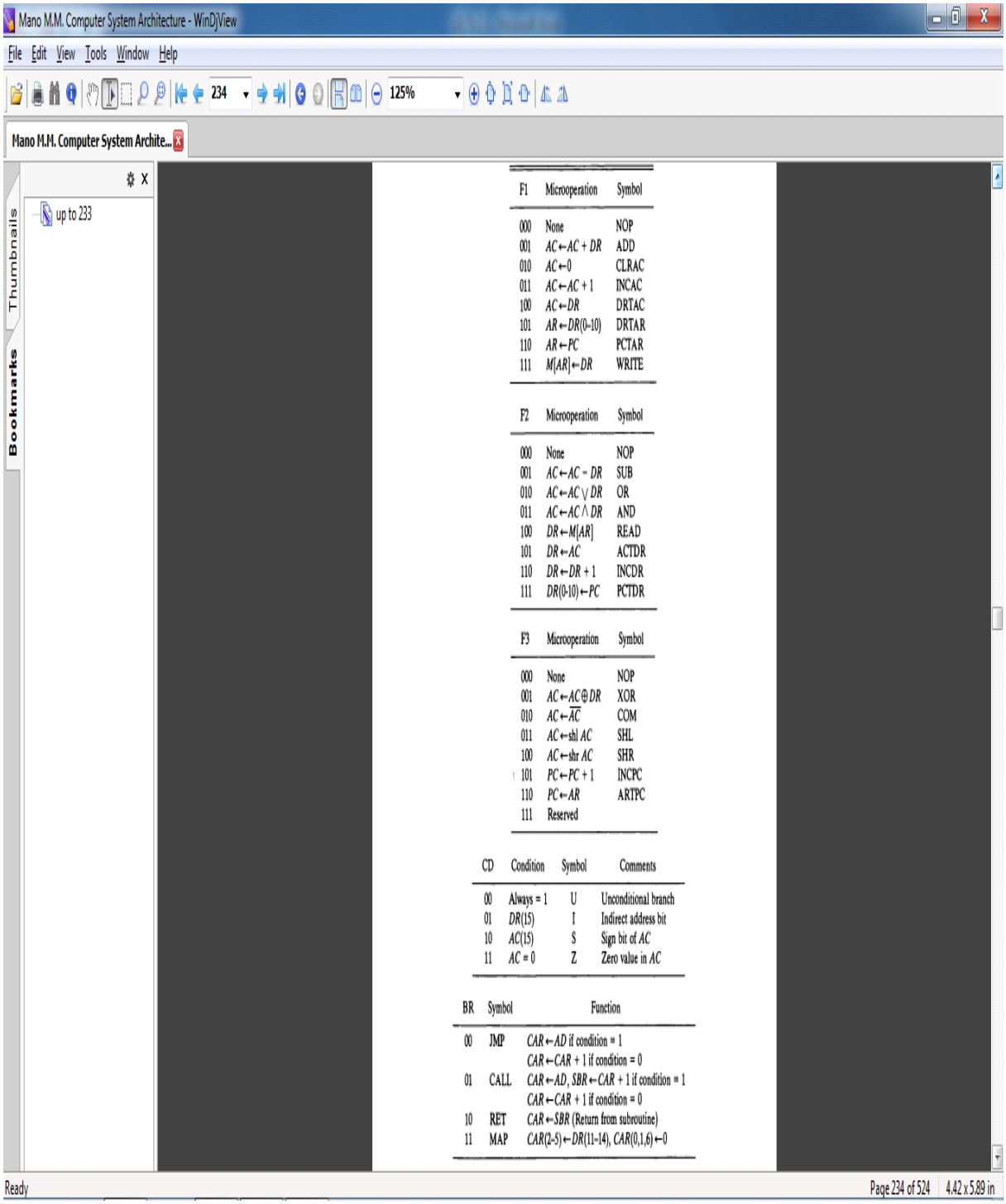


Microinstruction Format



The microinstruction format for the control memory is shown in Fig. 7-6. The20 bits of the microiristruction are divided into four functional parts. The three fields Fl, F2, and F3 specify microoperations for the computer.

The CD field selects status bit conditions. The BR field specifies the type of branch to be used. The AD field contains a branch address. The address field is seven bits wide, since the control memory has 128 = 27 words.



F1	Microoperation	Symbol
000	None	NOP
001	$AC \leftarrow AC + DR$	ADD
010	$AC \leftarrow 0$	CLRAC
011	$AC \leftarrow AC + 1$	INCAC
100	$AC \leftarrow DR$	DRTAC
101	$AR \leftarrow DR(0\text{-}10)$	DRTAR
110	$AR \leftarrow PC$	PCTAR
111	$M[AR] \leftarrow DR$	WRITE

F2	Microoperation	Symbol
000	None	NOP
001	$AC \leftarrow AC - DR$	SUB
010	$AC \leftarrow AC \vee DR$	OR
011	$AC \leftarrow AC \wedge DR$	AND
100	$DR \leftarrow M[AR]$	READ
101	$DR \leftarrow AC$	ACTDR
110	$DR \leftarrow DR + 1$	INCDR
111	$DR(0\text{-}10) \leftarrow PC$	PCTDR

F3	Microoperation	Symbol
000	None	NOP
001	$AC \leftarrow AC \oplus DR$	XOR
010	$AC \leftarrow \overline{AC}$	COM
011	$AC \leftarrow$ shl AC	SHL
100	$AC \leftarrow$ shr AC	SHR
101	$PC \leftarrow PC + 1$	INCPC
110	$PC \leftarrow AR$	ARTPC
111	Reserved	

CD	Condition	Symbol	Comments
00	Always = 1	U	Unconditional branch
01	$DR(15)$	I	Indirect address bit
10	$AC(15)$	S	Sign bit of AC
11	$AC = 0$	Z	Zero value in AC

BR	Symbol	Function
00	JMP	$CAR \leftarrow AD$ if condition = 1
		$CAR \leftarrow CAR + 1$ if condition = 0
01	CALL	$CAR \leftarrow AD$, $SBR \leftarrow CAR + 1$ if condition = 1
		$CAR \leftarrow CAR + 1$ if condition = 0
10	RET	$CAR \leftarrow SBR$ (Return from subroutine)
11	MAP	$CAR(2\text{-}5) \leftarrow DR(11\text{-}14)$, $CAR(0,1,6) \leftarrow 0$

The microoperations are subdivided into three fields of three bits each. The three bits in each field are encoded to specify seven distinct microoperations as listed in Table 7-1. This gives a total of 21 microoperations. No more than three microoperations can be chosen for a microinstruction, one from each field. If fewer than three microoperations are used, one or more of the fields will use the binary code 000 for no operation. As an illustration, a microinstruction can specify two simultaneous microoperations from F2 and F3 and none from Fl.

DR M[AR] with F2 = 100

and PC `PC + 1 with F3 = 101

The nine bits of the microoperation fields will then be 000 100 101.

When condition bits (CD) are used in conjunction with the BR (branch) field, it provides an unconditional branch operation.

The first condition is always a 1, so that a reference to CD = 00 (or the symbol U) will always find the condition to be true. When this condition is used in conjunction with the BR (branch) field, it provides an unconditional branch operation. The indirect bit I is available from bit 15 of DR after an instruction is read from memory. The sign bit of AC provides the next status bit. The zero value, symbolized by Z, is a binary variable whose value is equal to 1 if all the bits in AC are equal to zero.

The BR (branch) field consists of two bits. It is used, in conjunction with the address field AD, to choose the address of the next microinstruction. As shown in Table 7-1, when BR = 00, the control performs a jump QMP) operation (which is similar to a branch), and when BR = 01, it performs a call to subroutine (CALL) operation. The two operations are identical except that a call microinstruction stores the return address in the subroutine register SBR. The jump and call operations depend on the value of the CD field. If the status bit condition specified in the CD field is equal to 1, the next address in the AD field is transferred to the control address register CAR. Otherwise, CAR is incremented by 1.

The return from subroutine is accomplished with a BR field equal to 10. This causes the transfer of the return address from SBR to CAR. The mapping from the operation code bits of the instruction to an address for CAR is accomplished when the BR field is equal to 11. This mapping is as depicted in Fig. 7-3. The bits of the operation code are in DR(11-14) after an instruction is read from memory. Note that the last two conditions in the BR field are independent of the values in the CD and AD fields.

NEXT P235 (Symbolic Microinstructions)

*Symbolic Microinstructions

Each symbolic microinstruction is divided into five fields: label, microoperations, CD, BR, and AD.

* The Fetch Routine(and fetch & decode)

* Binary microprogram

Hardwired vs Microprogrammed control

There are two major types of control organization: hardwired control and microprogrammed control. In the hardwired organization, the control logic is implemented with gates, flip-flops, decoders, and other digital circuits. It has the advantage that it can be optimized to produce a fast mode of operation. In the microprogrammed organization, the control information is stored in a control memory. The control memory is programmed to initiate the required sequence of microoperations. A hardwired control, as the name implies, requires changes in the wiring among the various components if the design has to be modified or changed. In the microprogrammed control, any required changes or modifications can be done by updating the microprogram in control memory.

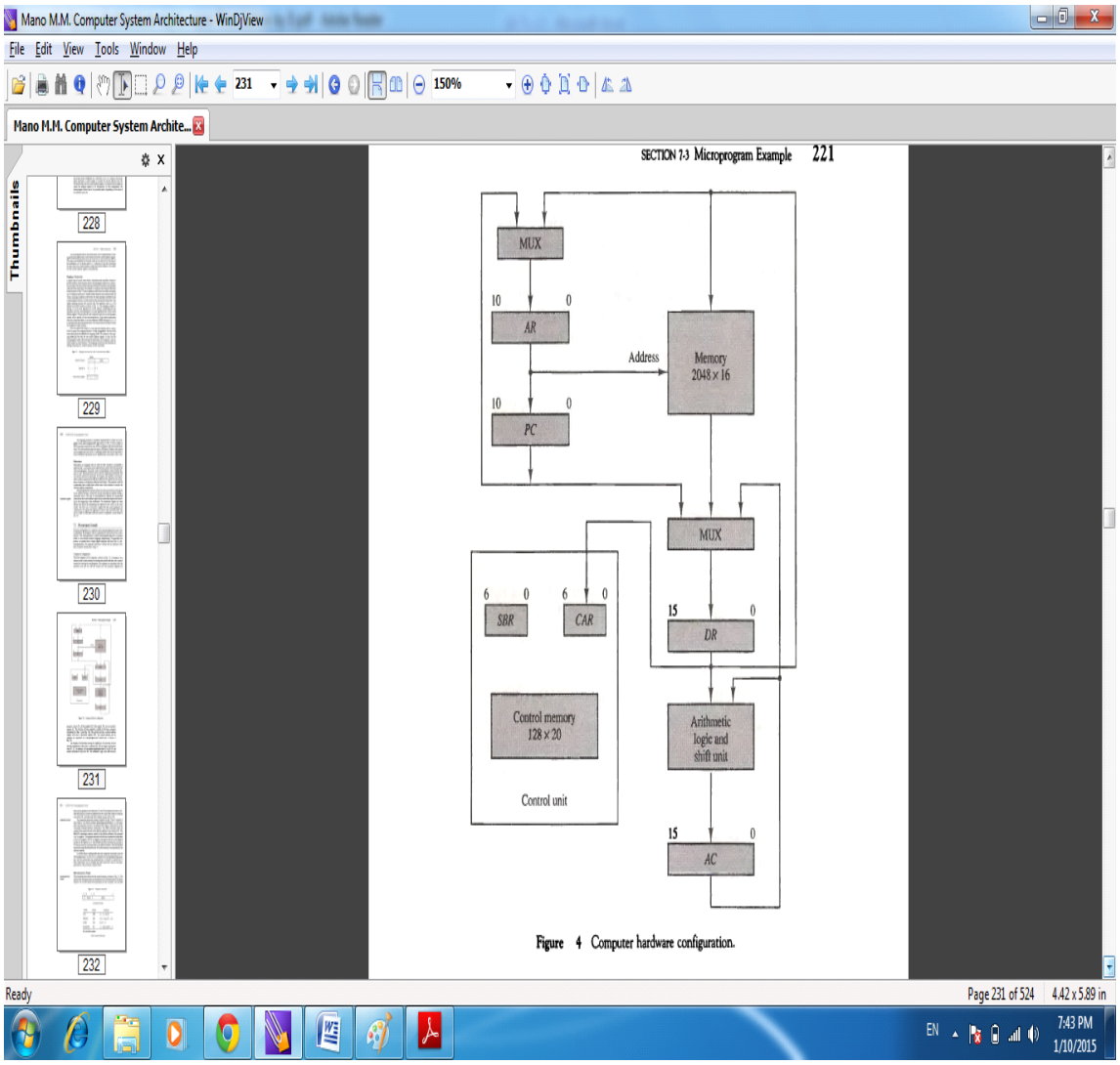
Subroutines

Subroutines are programs that are used by other routines to accomplish a particular task. A subroutine can be called from any point within the main body of the microprogram.

For example, the sequence of microoperations needed to generate the effective address of the operand for an instruction is common to all memory

reference instructions. This sequence could be a subroutine that is called from within many other routines to execute the effective address computation.

Microprograms that use subroutines must have a provision احتياط for storing the return address during a subroutine call and restoring the address during a subroutine return. This may be accomplished by placing the incrementedoutput from the control address register into a subroutine register and branching to the beginning of the subroutine. The subroutine register can then become the source for transferring the address for the return to the main routine.



Computer microprogram Example

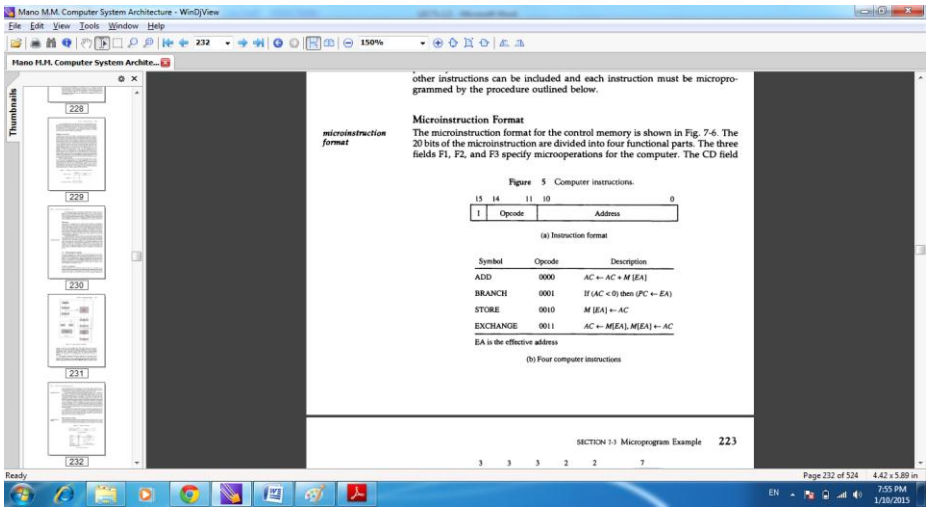
The block diagram of the computer shown consists of two memory units: a main memory for storing instructions and data, and a control memory for storing the microprogram. Four registers are associated with the processor unit and two with the control unit. The processor registers are program

counter PC, address register AR, data register DR, and accumulator register AC.

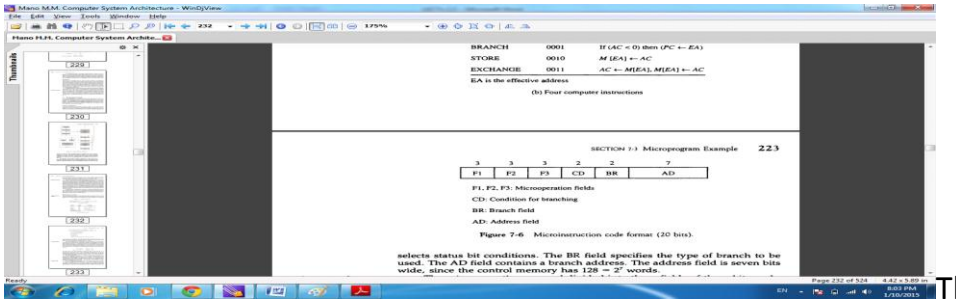
The control unit has a control address register CAR and a subroutine register SBR.

Computer instruction format

The computer instruction format is shown in fig 5(b), consists of three fields: a 1-bit held for indirect addressing symbolized by (I), a 4-bit operation code (opcode), and an 11-bit address field. Figure 5(b) lists four of the 16 possible memory-reference instructions. The ADD instruction adds the content of the operand found in the effective address to the content of AC. The BRANCH instruction causes a branch to the effective address if the operand in AC is negative. The program proceeds with the next consecutive instruction if AC is not negative. The AC is negative if its sign bit (the bit in the leftmost position of the register) is a 1. The STORE instruction transfers the content of AC into the memory word specified by the effective address. The EXCHANGE instruction swaps the data between AC and the memory word specified by the effective address.

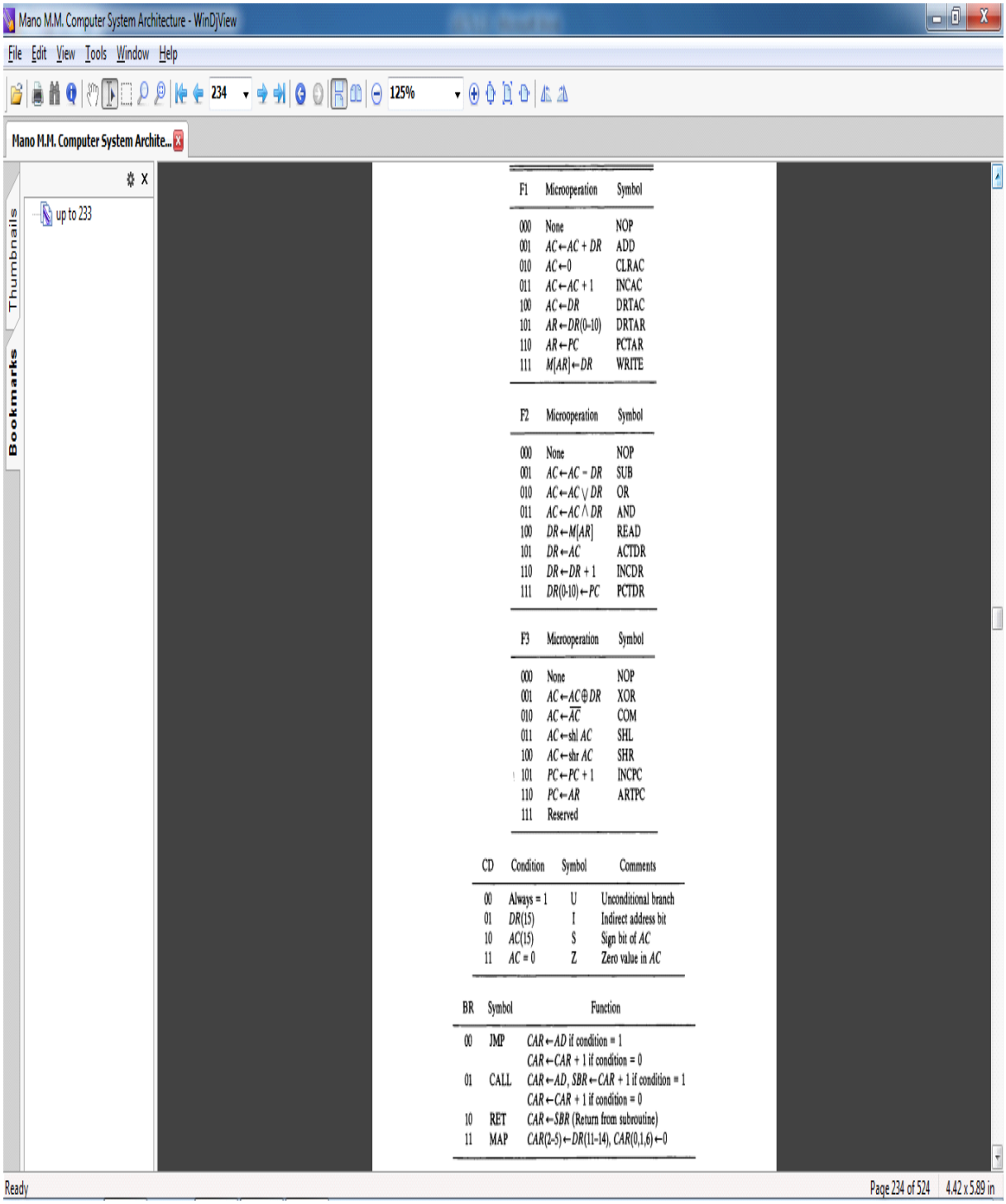


Microinstruction Format



The microinstruction format for the control memory is shown in Fig. 7-6. The20 bits of the microiristruction are divided into four functional parts. The three fields Fl, F2, and F3 specify microoperations for the computer.

The CD field selects status bit conditions. The BR field specifies the type of branch to be used. The AD field contains a branch address. The address field is seven bits wide, since the control memory has 128 = 27 words.



F1	Microoperation	Symbol
000	None	NOP
001	$AC \leftarrow AC + DR$	ADD
010	$AC \leftarrow 0$	CLRAC
011	$AC \leftarrow AC + 1$	INCAC
100	$AC \leftarrow DR$	DRTAC
101	$AR \leftarrow DR(0\text{-}10)$	DRTAR
110	$AR \leftarrow PC$	PCTAR
111	$M[AR] \leftarrow DR$	WRITE

F2	Microoperation	Symbol
000	None	NOP
001	$AC \leftarrow AC - DR$	SUB
010	$AC \leftarrow AC \vee DR$	OR
011	$AC \leftarrow AC \wedge DR$	AND
100	$DR \leftarrow M[AR]$	READ
101	$DR \leftarrow AC$	ACTDR
110	$DR \leftarrow DR + 1$	INCDR
111	$DR(0\text{-}10) \leftarrow PC$	PCTDR

F3	Microoperation	Symbol
000	None	NOP
001	$AC \leftarrow AC \oplus DR$	XOR
010	$AC \leftarrow \overline{AC}$	COM
011	$AC \leftarrow$ shl AC	SHL
100	$AC \leftarrow$ shr AC	SHR
101	$PC \leftarrow PC + 1$	INCPC
110	$PC \leftarrow AR$	ARTPC
111	Reserved	

CD	Condition	Symbol	Comments
00	Always = 1	U	Unconditional branch
01	$DR(15)$	I	Indirect address bit
10	$AC(15)$	S	Sign bit of AC
11	$AC = 0$	Z	Zero value in AC

BR	Symbol	Function
00	JMP	$CAR \leftarrow AD$ if condition = 1
		$CAR \leftarrow CAR + 1$ if condition = 0
01	CALL	$CAR \leftarrow AD$, $SBR \leftarrow CAR + 1$ if condition = 1
		$CAR \leftarrow CAR + 1$ if condition = 0
10	RET	$CAR \leftarrow SBR$ (Return from subroutine)
11	MAP	$CAR(2\text{-}5) \leftarrow DR(11\text{-}14)$, $CAR(0,1,6) \leftarrow 0$

The microoperations are subdivided into three fields of three bits each. The three bits in each field are encoded to specify seven distinct microoperations as listed in Table 7-1. This gives a total of 21 microoperations. No more than three microoperations can be chosen for a microinstruction, one from each field. If fewer than three microoperations are used, one or more of the fields will use the binary code 000 for no operation. As an illustration, a microinstruction can specify two simultaneous microoperations from F2 and F3 and none from Fl.

DR M[AR] with F2 = 100

and PC `PC + 1 with F3 = 101

The nine bits of the microoperation fields will then be 000 100 101.

When condition bits (CD) are used in conjunction with the BR (branch) field, it provides an unconditional branch operation.

The first condition is always a 1, so that a reference to CD = 00 (or the symbol U) will always find the condition to be true. When this condition is used in conjunction with the BR (branch) field, it provides an unconditional branch operation. The indirect bit I is available from bit 15 of DR after an instruction is read from memory. The sign bit of AC provides the next status bit. The zero value, symbolized by Z, is a binary variable whose value is equal to 1 if all the bits in AC are equal to zero.

The BR (branch) field consists of two bits. It is used, in conjunction with the address field AD, to choose the address of the next microinstruction. As shown in Table 7-1, when BR = 00, the control performs a jump QMP) operation (which is similar to a branch), and when BR = 01, it performs a call to subroutine (CALL) operation. The two operations are identical except that a call microinstruction stores the return address in the subroutine register SBR. The jump and call operations depend on the value of the CD field. If the status bit condition specified in the CD field is equal to 1, the next address in the AD field is transferred to the control address register CAR. Otherwise, CAR is incremented by 1.

The return from subroutine is accomplished with a BR field equal to 10. This causes the transfer of the return address from SBR to CAR. The mapping from the operation code bits of the instruction to an address for CAR is accomplished when the BR field is equal to 11. This mapping is as depicted in Fig. 7-3. The bits of the operation code are in DR(11-14) after an instruction is read from memory. Note that the last two conditions in the BR field are independent of the values in the CD and AD fields.

NEXT P235 (Symbolic Microinstructions)

*Symbolic Microinstructions

Each symbolic microinstruction is divided into five fields: label, microoperations, CD, BR, and AD.

* The Fetch Routine(and fetch & decode)

* Binary microprogram

Hardwired vs Microprogrammed control

There are two major types of control organization: hardwired control and microprogrammed control. In the hardwired organization, the control logic is implemented with gates, flip-flops, decoders, and other digital circuits. It has the advantage that it can be optimized to produce a fast mode of operation. In the microprogrammed organization, the control information is stored in a control memory. The control memory is programmed to initiate the required sequence of microoperations. A hardwired control, as the name implies, requires changes in the wiring among the various components if the design has to be modified or changed. In the microprogrammed control, any required changes or modifications can be done by updating the microprogram in control memory.

Pipeline and Vector Processing

Parallel Processing

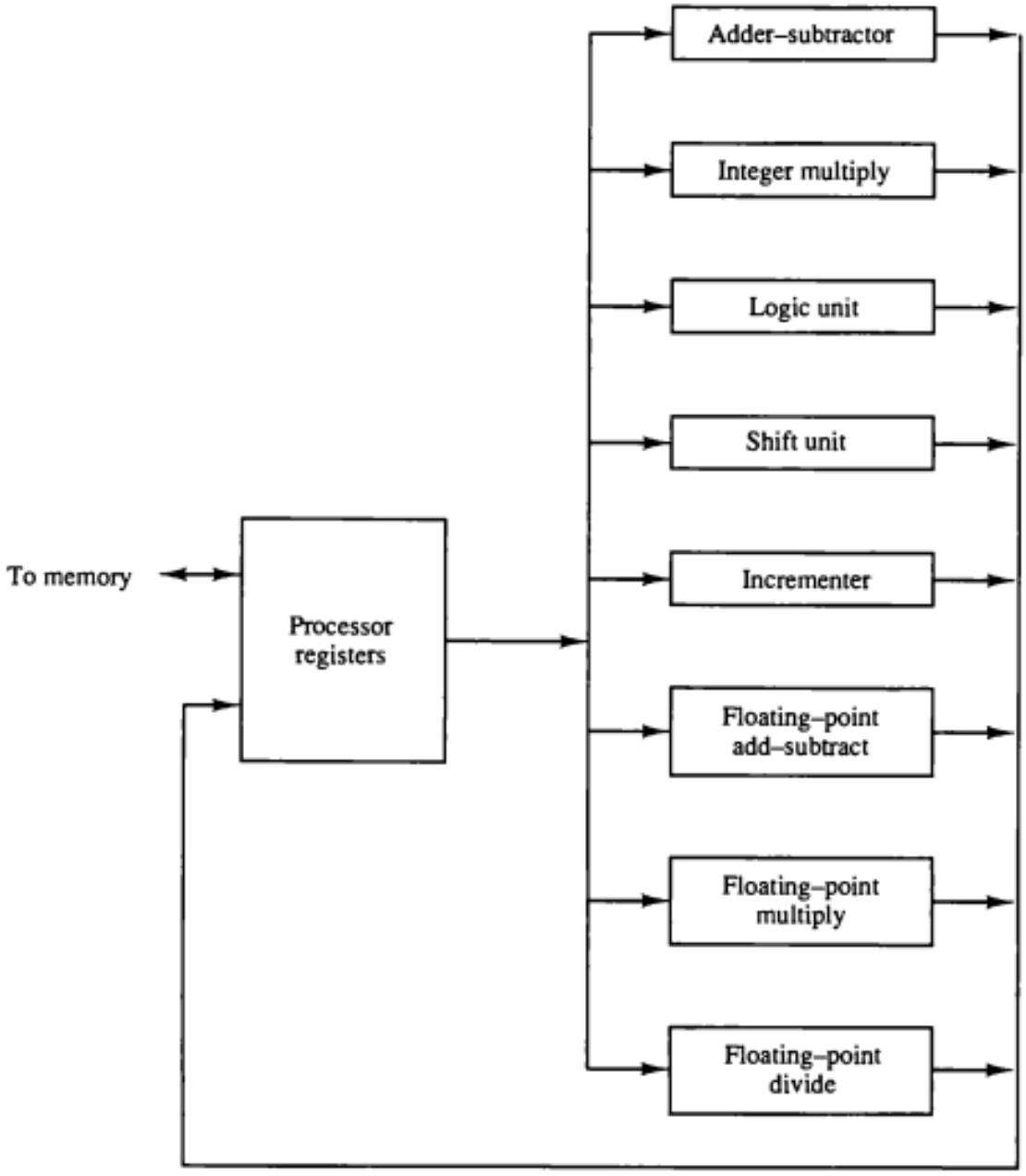
Parallel processing is a term used to denote a large class of techniques that are used to provide simultaneous data-processing tasks for the purpose of increasing the computational speed of a computer system. Instead of processing each instruction sequentially as in a conventional computer, a parallel processing system is able to perform concurrent data processing to achieve faster execution time. For example, while an instruction is being executed in the ALU, the next instruction can be read from memory. The

system may have two or more ALUs and be able to execute two or more instructions at the same time.

Furthermore, the system may have two or more processors operating concurrently. The purpose of parallel processing is to speed up the computer processing capability and increase its throughput, that is, the amount of processing that can be accomplished during a given interval of time. The amount of hardware increases with parallel processing, and with it, the cost of the system increases. However, technological developments have reduced hardware costs to the point where parallel processing techniques are economically feasible.

Parallel processing can be viewed from various levels of complexity. At the lowest level, we distinguish between parallel and serial operations by the type of registers used. Shift registers operate in serial fashion one bit at a time, while registers with parallel load operate with all the bits of the word simultaneously. Parallel processing at a higher level of complexity can be achieved by having a multiplicity of functional units that perform identical or different operations simultaneously. Parallel processing is established by distributing the data among the multiple functional units. For example, the arithmetic, logic, and shift operations can be separated into three units and the operands diverted to each unit under the supervision of a control unit.

Figure below shows one possible way of separating the execution unit into eight functional units operating in parallel. The operands in the registers are applied to one of the units depending on the operation specified by the instruction associated with the operands.



The operation performed in each functional unit is indicated in each block of the diagram. The adder and integer multiplier perform the arithmetic operations with integer numbers. The floating-point operations are separated into three circuits operating in parallel. The logic, shift, and increment operations can be performed concurrently on different data. All units are independent of each other, so one number can be shifted while another number is being incremented. A multifunctional organization is usually

associated with a complex control unit to coordinate all the activities among the various components.

There are a variety of ways that parallel processing can be classified. It can be considered from the internal organization of the processors, from the interconnection structure between processors, or from the flow of information through the system. One classification introduced by M. J. Flynn considers the organization of a computer system by the number of instructions and data items that are manipulated simultaneously. The normal operation of a computer is to fetch instructions from memory and execute them in the processor.

The sequence of instructions read from memory constitutes an instruction stream. The operations performed on the data in the processor constitutes a data stream. Parallel processing may occur in the instruction stream, in the data stream, or in both. Flynn's classification divides computers into four major groups as follows:

Single instruction stream, single data stream (SISD)

Single instruction stream, multiple data stream (SIMD)

Multiple instruction stream, single data stream (MISD)

Multiple instruction stream, multiple data stream (MIMD)

SISD represents the organization of a single computer containing a control unit, a processor unit, and a memory unit. Instructions are executed sequentially and the system may or may not have internal parallel processing capabilities. Parallel processing in this case may be achieved by means of multiple functional units or by pipeline processing.

SIMD represents an organization that includes many processing units under the supervision of a common control unit. All processors receive the same instruction from the control unit but operate on different items of data. The shared memory unit must contain multiple modules so that it can communicate with all the processors simultaneously. MISD structure is only of

theoretical interest since no practical system has been constructed using this organization.

MIMD organization refers to a computer system capable of processing several programs at the same time. Most multiprocessor and multicomputer systems can be classified in this category. Flynn's classification depends on the distinction between the performance of the control unit and the data-processing unit. It emphasizes the behavioral characteristics of the computer system rather than its operational and structural interconnections. One type of parallel processing that does not fit Flynn's classification is pipelining. The only two categories used from this classification are SIMD array processors and MIMD.

In this chapter we consider parallel processing under the following main

topics:

1. Pipeline processing

2. Vector processing

3. Array processors

Pipeline processing is an implementation technique where arithmetic suboperations or the phases of a computer instruction cycle overlap in execution.

Vector processing deals with computations involving large vectors and matrices. Array processors perform computations on large arrays of data.

Pipelining

Pipelining is a technique of decomposing a sequential process into suboperations, with each subprocess being executed in a special dedicated segment that operates concurrently with all other segments.

A pipeline can be visualized as a collection of processing segments through which binary information flows.

Each segment performs partial processing dictated by the way the task is partitioned. The result obtained from the computation in each segment is transferred to the next segment in the pipeline. The final result is obtained after the data have passed through all segments. The name "pipeline" implies a

flow of information analogous to an industrial assembly line. It is characteristic of pipelines that several computations can be in progress in distinct segments at the same time. The overlapping of computation is made possible by associating a register with each segment in the pipeline. The registers provide isolation between each segment so that each can operate on distinct data simultaneously.

Perhaps the simplest way of viewing the pipeline structure is to imagine that each segment consists of an input register followed by a combinational circuit. The register holds the data and the combinational circuit performs the suboperation in the particular segment. The output of the combinational circuit in a given segment is applied to the input register of the next segment. A clock is applied to all registers after enough time has elapsed to perform all segment activity. In this way the information flows through the pipeline one step at a time.

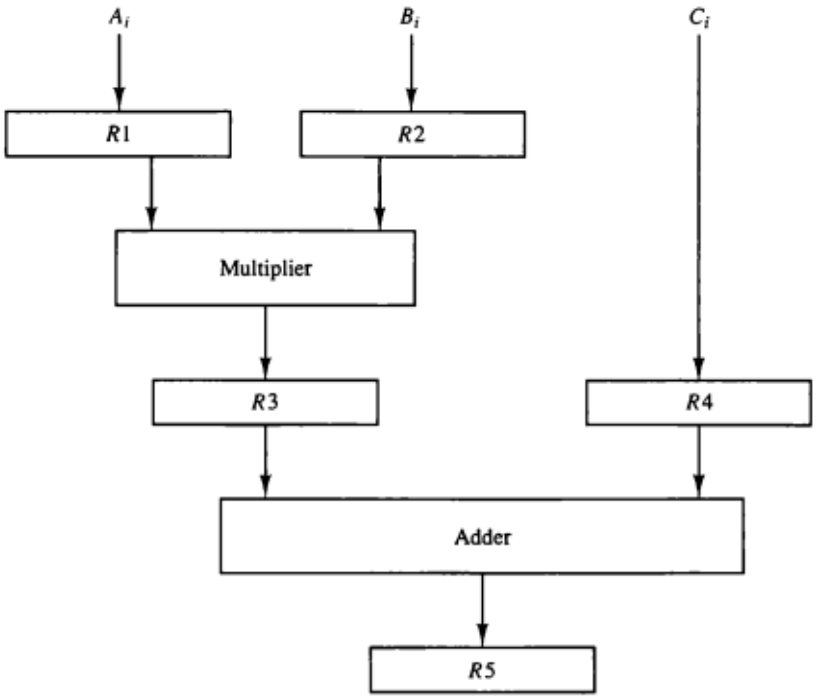
The pipeline organization will be demonstrated by means of a simple example. Suppose that we want to perform the combined multiply and add operations with a stream of numbers.

$A_i * B_i + C_i$ for $i = 1, 2, 3, \ldots, 7$

Each suboperation is to be implemented in a segment within a pipeline. Each segment has one or two registers and a combinational circuit as shown in Fig. 2. Rl through R5 are registers that receive new data with every clock pulse. The multiplier and adder are combinational circuits. The suboperations performed in each segment of the pipeline are as follows:

$R1 \leftarrow A_i, \quad R2 \leftarrow B_i$	Input A_i and B_i
$R3 \leftarrow R1 * R2, \quad R4 \leftarrow C_i$	Multiply and input C_i
$R5 \leftarrow R3 + R4$	Add C_i to product

The five registers are loaded with new data every clock pulse. The effect of each clock is shown in Table 1. The first clock pulse transfers Ai and Bi into R1 and R2.



Clock Pulse Number	Segment 1		Segment 2		Segment 3
	$R1$	$R2$	$R3$	$R4$	$R5$
1	A_1	B_1	—	—	—
2	A_2	B_2	$A_1 * B_1$	C_1	—
3	A_3	B_3	$A_2 * B_2$	C_2	$A_1 * B_1 + C_1$
4	A_4	B_4	$A_3 * B_3$	C_3	$A_2 * B_2 + C_2$
5	A_5	B_5	$A_4 * B_4$	C_4	$A_3 * B_3 + C_3$
6	A_6	B_6	$A_5 * B_5$	C_5	$A_4 * B_4 + C_4$
7	A_7	B_7	$A_6 * B_6$	C_6	$A_5 * B_5 + C_5$
8	—	—	$A_7 * B_7$	C_7	$A_6 * B_6 + C_6$
9	—	—	—	—	$A_7 * B_7 + C_7$

The second clock pulse transfers the product of Rl and R2 into R3 and Q into R4. The same clock pulse transfers A2 and B2 into Rl and R2. The third clock pulse operates on all three segments simultaneously. It places A3 and B3 into Rl and R2, transfers the product of Rl and R2 into R3, transfers C2 into R4, and places the sum of R3 and R4 into R5. It takes three clock pulses to fill up the pipe and retrieve the first output from R5. From there on, each clock produces a new output and moves the data one step down the pipeline. This happens as long as new input data flow into the system. When no more input data are available, the clock must continue until the last output emerges out of the pipeline.

Arithmetic Pipeline

Pipeline arithmetic units are usually found in very high speed computers. They are used to implement floating-point operations, multiplication of fixed-point numbers, and similar computations encountered in scientific problems. A pipeline multiplier is essentially an array multiplier as described in Figure below, with special adders designed to minimize the carry propagation time through the partial products. Floating-point operations are easily decomposed into suboperations.

We will now show an example of a pipeline unit for floating-point addition and subtraction. The inputs to the floating-point adder pipeline are two normalized floating-point binary numbers.

$$X = A \times 2^a$$
$$Y = B \times 2^b$$

A and B are two fractions that represent the mantissas and a and b are the

exponents. The floating-point addition and subtraction can be performed in

four segments, as shown in Fig. 9-6. The registers labeled R are placed between

the segments to store intermediate results. The suboperations that are

performed in the four segments are:

1. Compare the exponents.

2. Align the mantissas.

3. Add or subtract the mantissas.

4. Normalize the result.

This follows the procedure outlined in the flowchart of Fig. 3 but with some

variations that are used to reduce the execution time of the suboperations.

The exponents are compared by subtracting them to determine their difference.

The larger exponent is chosen as the exponent of the result. The exponent difference determines how many times the mantissa associated with the smaller exponent must be shifted to the right. This produces an alignment of the two mantissas. It should be noted that the shift must be designed as a combinational circuit to reduce the shift time. The two mantissas are added or subtracted in segment 3. The result is normalized in segment 4. When an overflow occurs, the mantissa of the sum or difference is shifted right and the exponent incremented by one. If an underflow occurs, the number of leading zeros in the mantissa determines the number of left shifts in the mantissa and the number that must be subtracted from the exponent.

The following numerical example may clarify the suboperations performed in each segment. For simplicity, we use decimal numbers, although Fig. 9-6 refers to binary numbers. Consider the two normalized floating-point numbers:

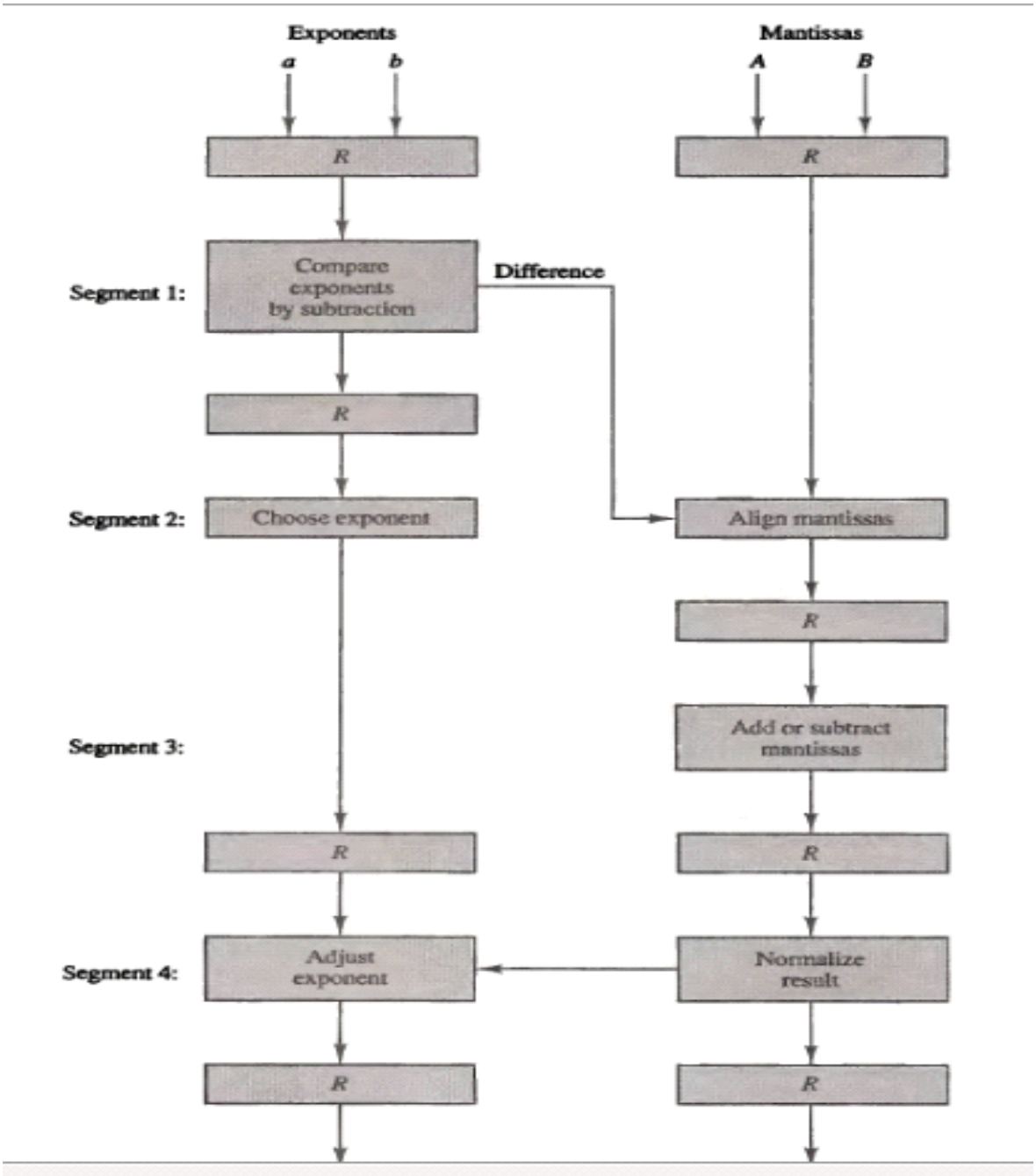
$$X = 0.9504 \times 10^3$$
$$Y = 0.8200 \times 10^2$$

The two exponents are subtracted in the first segment to obtain 3-2 = 1. The larger exponent 3 is chosen as the exponent of the result. The next segment shifts the mantissa of Y to the right to obtain

$$X = 0.9504 \times 10^3$$
$$Y = 0.0820 \times 10^3$$

This aligns the two mantissas under the same exponent. The addition of the two mantissas in segment 3 produces the sum

$$Z = 1.0324 \times 10^3$$



The sum is adjusted by normalizing the result so that it has a fraction with a nonzero first digit. This is done by shifting the mantissa once to the right and incrementing the exponent by one to obtain the normalized sum.

$$Z = 0.10324 \times 10^4$$