UNIT II

1.1 Micro programmed control:

Hardwired Control Unit:

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

Micro programmed control unit:

A control unit whose binary control variable are stored in memory is called a micro programmed control unit.

Dynamic microprogramming:

A more advanced development known as dynamic microprogramming permits a microprogram to be loaded initially from an auxiliary memory such as a magnetic disk. Control units that use dynamic microprogramming employ

a writable control memory. This type of memory can be used for writing.

Control Memory:

Control Memory is the storage in the microprogrammed control unit to store the microprogram.

Writeable Control Memory:

Control Storage whose contents can be modified, allow the change in microprogram and Instruction set can be changed or modified is referred as Writeable Control Memory.

Control Word:

The control variables at any given time can be represented by a control word string of 1 's and 0's called a control word.

Microoperation, Microinstruction, Micro program, Microcode.

Microoperations:

- In computer central processing units, micro-operations (also known as a micro-ops or μops) are detailed low-level instructions used in some designs to implement complex machine instructions (sometimes termed macro-instructions in this context).

Micro instruction:

- A symbolic microprogram can be translated into its binary equivalent by means of an assembler.
- Each line of the assembly language microprogram defines a symbolic microinstruction.
- Each symbolic microinstruction is divided into five fields: label, microoperations, CD, BR, and AD.

Micro program:

- A sequence of microinstructions constitutes a microprogram.
- Since alterations of the microprogram are not needed once the control unit is in operation,the control memory can be a read-only memory (ROM).
- ROM words are made permanent during the hardware production of the unit.
- The use of a micro program involves placing all control variables in words of ROM for use by the control unit through successive read operations.
- The content of the word in ROM at a given address specifies a microinstruction.

Organization of micro programmed control unit

- The general configuration of a micro-programmed control unit is demonstrated in the block diagram of Figure 4.1.
- The control memory is assumed to be a ROM, within which all control information is permanently stored.

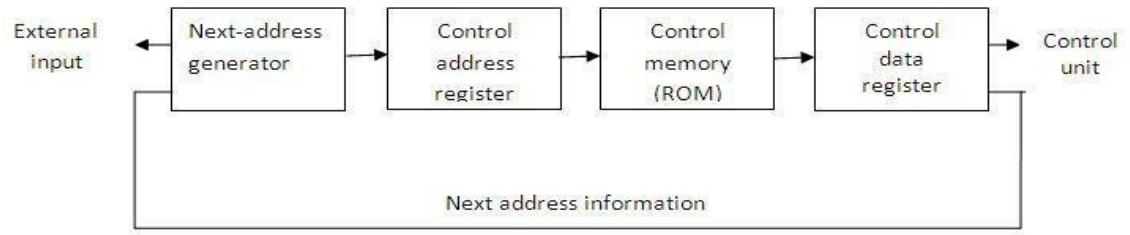


figure 4.1: Micro-programmed control organization

- 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.
- The location of the next microinstruction may be the one next in sequence, or it may be located somewhere else in the control memory.

- While the microoperations are being executed, the next address is computed in the next address generator circuit and then transferred into the control address register to read the next microinstruction.
- Thus a microinstruction contains bits for initiating microoperations in the data processor part and bits that determine the address sequence for the control memory.
- The next address generator is sometimes called a micro-program sequencer, as it determines the address sequence that is read from control memory.
- Typical functions of a micro-program sequencer are incrementing the control address register by one, loading into the control address register an address from control

memory, transferring an external address, or loading an initial address to start the control operations.

- The control data register holds the present microinstruction while the next address is computed and read from memory.
- The data register is sometimes called a pipeline register.
- It allows the execution of the microoperations specified by the control word simultaneously with the generation of the next microinstruction.
- This configuration requires a two-phase clock, with one clock applied to the address register and the other to the data register.
- The main advantage of the micro programmed 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.

Address Sequencing:

- Microinstructions are stored in control memory in groups, with each group specifying a routine.
- To appreciate the address sequencing in a micro-program control unit, let us specify the steps that the control must undergo during the execution of a single computer instruction.

Step-1:

- An initial address is loaded into the control address register when power is turned on in the computer.
- This address is usually the address of the first microinstruction that activates the instruction fetch routine.
- The fetch routine may be sequenced by incrementing the control address register through the rest of its microinstructions.
- At the end of the fetch routine, the instruction is in the instruction register of the computer.

Step-2:

- The control memory next must go through the routine that determines the effective address of the operand.
- A machine instruction may have bits that specify various addressing modes, such as indirect address and index registers.
- The effective address computation routine in control memory can be reached through a branch microinstruction, which is conditioned on the status of the mode bits of the instruction.
- When the effective address computation routine is completed, the address of the operand is available in the memory address register.

Step-3:

- The next step is to generate the microoperations that execute the instruction fetched from memory.
- The microoperation steps to be generated in processor registers depend on the operation code part of the instruction.
- Each instruction has its own micro-program routine stored in a given location of control memory.
- 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.

Step-4:

- Once the required routine is reached, the microinstructions that execute the instruction may be sequenced by incrementing the control address register.
- Micro-programs that employ subroutines will require an external register for storing the return address.
- Return addresses cannot be stored in ROM because the unit has no writing capability.
- When the execution of the instruction is completed, control must return to the fetch routine.
- This is accomplished by executing an unconditional branch microinstruction to the first address of the fetch routine.

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.

selection of address for control memory

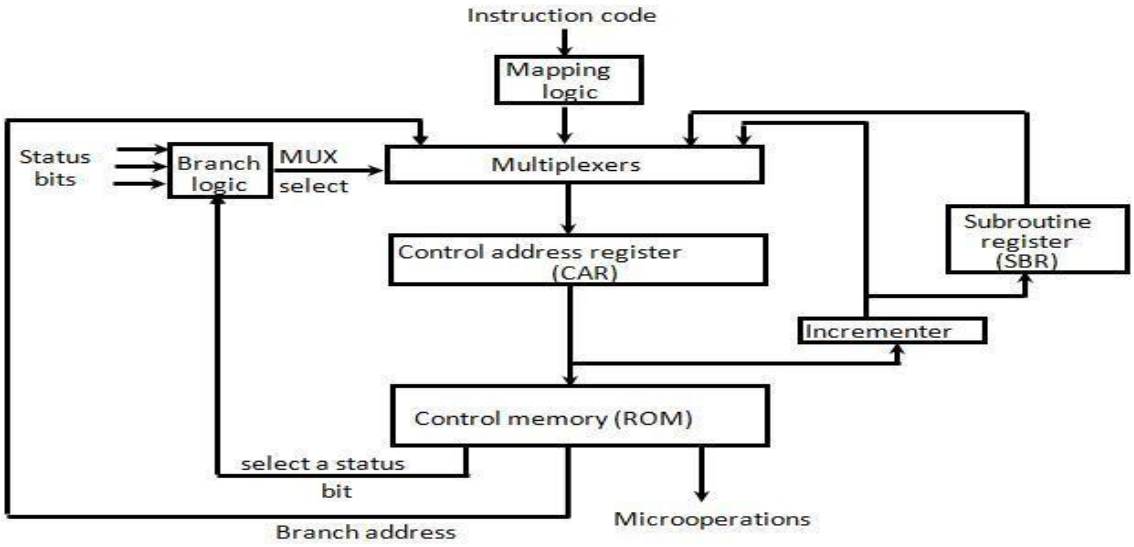


Figure 4.2: Selection of address for control memory

- Above figure 4.2 shows a block diagram of a control memory and the associated hardware needed for selecting the next microinstruction address.
- The microinstruction in control memory contains a set of bits to initiate microoperations in computer registers and other bits to specify the method by which the next address is obtained.
- 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 micro-program wishes to return from the subroutine.
- The branch logic of figure 4.2 provides decision-making capabilities in the control unit.
- 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.

- 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.
- A 1 output in the multiplexer generates a control signal to transfer the branch address from the microinstruction into the control address register.
- A 0 output in the multiplexer causes the address register to be incremented.

Micro programme Example:

Mapping of an Instruction

- A special type of branch exists when a microinstruction specifies a branch to the first word in control memory where a microprogram routine for an instruction is located.
- The status bits for this type of branch are the bits in the operation code part of the instruction.For example, a computer with a simple instruction format as shown in figure 4.3 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.
- One simple mapping process that converts the 4-bit operation code to a 7-bit address for control memory is shown in figure 4.3.
- 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.

- This provides for each computer instruction a microprogram routine with a capacity of four microinstructions.
- If the routine needs more than four microinstructions, it can use addresses 1000000 through 1111111. If it uses fewer than four microinstructions, the unused memory locations would be available for other routines.

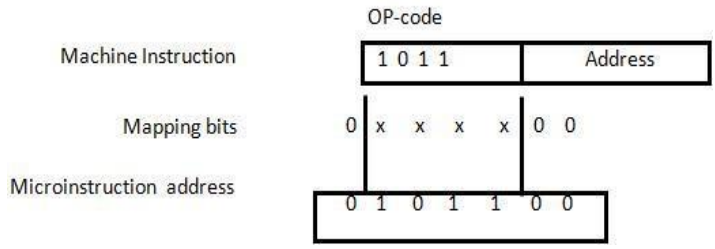


Figure 4.3: Mapping from instruction code to microinstruction address

- One can extend this concept to a more general mapping rule by using a ROM to specify the mapping function.
- The contents of the mapping ROM give the bits for the control address register.
- In this way the microprogram routine that executes the instruction can be placed in any desired location in control memory.
- The mapping concept provides flexibility for adding instructions for control memory as the need arises.

Computer Hardware Configuration

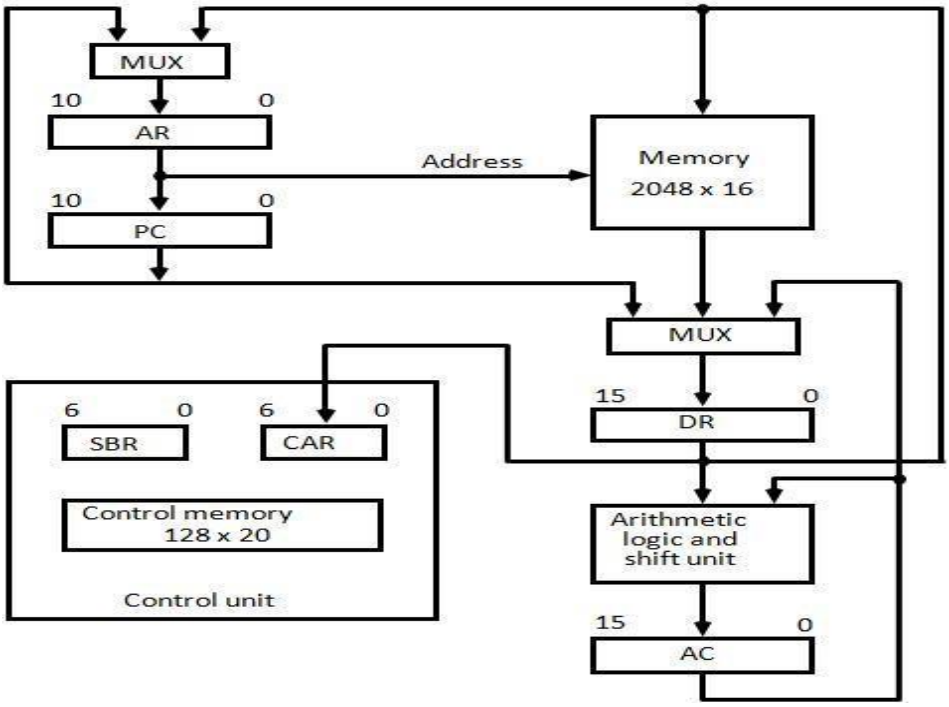


Figure 4.4: Computer hardware configuration

The block diagram of the computer is shown in Figure 4.4. It consists of

1. Two memory units:

   Main memory -> for storing instructions and data, and

   Control memory -> for storing the microprogram.

2. Six Registers:

   Processor unit register: AC(accumulator),PC(Program Counter), AR(Address Register),DR(Data Register)

   Control unit register: CAR (Control Address Register), SBR(Subroutine Register)

3. Multiplexers:

   The transfer of information among the registers in the processor is done through multiplexers rather than a common bus.

4. ALU:

   The arithmetic, logic, and shift unit performs microoperations with data from AC and DR and places the result in AC.

   - DR can receive information from AC, PC, or memory.
   - AR can receive information from PC or DR.
   - PC can receive information only from AR.
   - Input data written to memory come from DR, and data read from memory can go only to DR.

Microinstruction Format

The microinstruction format for the control memory is shown in figure 4.5. The 20 bits of the microinstruction are divided into four functional parts as follows:

1. The three fields F1, F2, and F3 specify microoperations for the computer.
   The microoperations are subdivided into three fields of three bits each. The three bits in each field are encoded to specify seven distinct microoperations. This gives a total of 21 microoperations.
2. The CD field selects status bit conditions.
3. The BR field specifies the type of branch to be used.
4. The AD field contains a branch address. The address field is seven bits wide, since the control memory has 128 = 27 words.

Figure 4.5: Microinstruction Format

- As an example, a microinstruction can specify two simultaneous microoperations fromF2 and F3 and none from F1.

  DR --- M[AR] with F2 = 100

  PC --- PC + 1 with F3 = 101

- The nine bits of the microoperation fields will then be 000 100 101.
- The CD (condition) field consists of two bits which are encoded to specify four status bit conditions as listed in Table 4.1.

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

Table 4.1: Condition Field

- 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 shown in Table 4.2.

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

Table 4.2: Branch Field

Symbolic Microinstruction:

- Each line of the assembly language microprogram defines a symbolic microinstruction.
- Each symbolic microinstruction is divided into five fields: label, microoperations, CD, BR, and AD. The fields specify the following Table 4.3.

1. Label	The label field may be empty or it may specify a symbolic address. A label is terminated with a colon (:).
2. Microoperations	It consists of one, two, or three symbols, separated by commas, from those defined in Table 5.3. There may be no more than one symbol from each F field. The NOP symbol is used when the microinstruction has no microoperations. This will be translated by the assembler to nine zeros.
3. CD	The CD field has one of the letters U, I, S, or Z.
4. BR	The BR field contains one of the four symbols defined in Table 5.2.
5. AD	The AD field specifies a value for the address field of the microinstruction in one of three possible ways: i. With a symbolic address, this must also appear as a label. ii. With the symbol NEXT to designate the next address in sequence. iii. When the BR field contains a RET or MAP symbol,the AD field is left empty and is converted to seven zeros by the assembler

Table 4.3: Symbolic Microinstruction

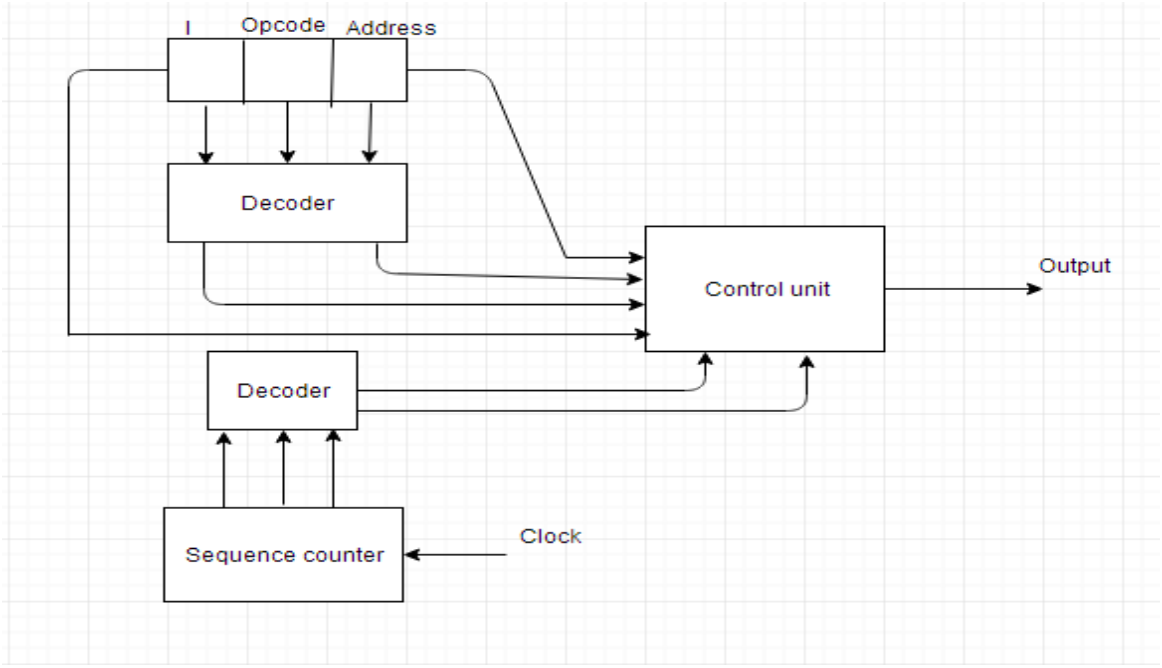
Design of Control Unit:

- Control unit generates timing and control signals for the operations of the computer.
- The control unit communicates with ALU and main memory.
- It also controls the transmission between processor, memory and the various peripherals. It also instructs the ALU which operation has to be performed on data.

Control unit can be designed by two methods which are given below:

Hardwired Control Unit:

- It is implemented with the help of gates, flip flops, decoders etc. in the hardware.
- The inputs to control unit are the instruction register, flags, timing signals etc.
- This organization can be very complicated if we have to make the control unit large. If the design has to be modified or changed, all the combinational circuits have to be modified which is a very difficult task.



Microprogrammed Control Unit:

- It is implemented by using programming approach.
- A sequence of micro operations is carried out by executing a program consisting of micro-instructions.
- In this organization any modifications or changes can be done by updating the micro program in the control memory by the programmer.

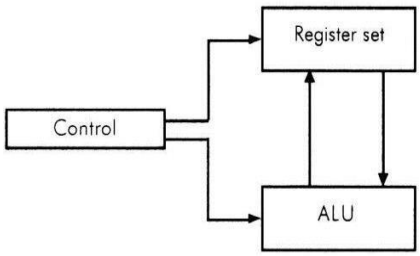
. **Difference between Hardwired Control and Microprogrammed Control:**

Hardwired Control	Microprogrammed Control
Technology is circuit based.	Technology is software based.
It is implemented through flip-flops, gates, decoders etc.	Microinstructions generate signals to control the execution of instructions.
Fixed instruction format.	Variable instruction format (16-64 bits per instruction).
Instructions are register based.	Instructions are not register based.
ROM is not used.	ROM is used.
It is used in RISC.	It is used in CISC.
Faster decoding.	Slower decoding.
Difficult to modify.	Easily modified.
Chip area is less.	Chip area is large.

2.2 Central Processing Unit

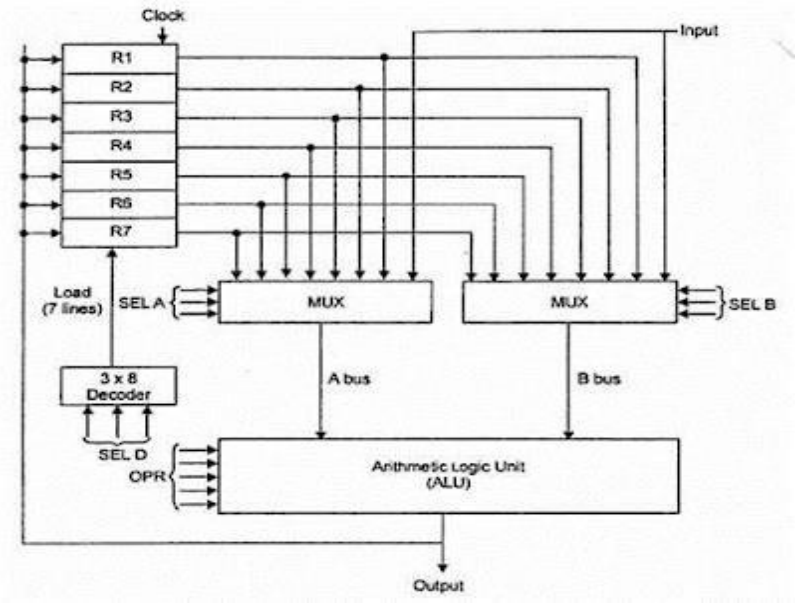
General Register Organization:

The Central Processing Unit (CPU) is called the brain of the computer that performs data processing operations. Figure 3.1 shows the three major parts of CPU.



- Intermediate data is stored in the register set during the execution of the instructions.
- The microoperations required for executing the instructions are performed by the arithmetic logic unit whereas the control unit takes care of transfer of information among the registers and guides the ALU.

- The control unit services the transfer of information among the registers and instructs the ALU about which operation is to be performed.
- The computer instruction set is meant for providing the specifications for the design of the CPU. The design of the CPU largely, involves choosing the hardware for implementing the machine instructions.
- The need for memory locations arises for storing pointers, counters, return address, temporary results and partial products.
- Memory access consumes the most of the time off an operation in a computer. It is more convenient and more efficient to store these intermediate values in processor registers.
- A common bus system is employed to contact registers that are included in the CPU in a large number.
- Communications between registers is not only for direct data transfer but also for performing various micro-operations.
- A bus organization for such CPU register shown in Figure 3.2, is connected to two multiplexers (MUX) to form two buses A and B. The selected lines in each multiplexers select one register of the input data for the particular bus.



EXAMPLE:

- To perform the operation ***R3 = R1+R2*** We have to provide following binary selection variable to the select inputs.

1. **SEL A** : **001** -To place the contents of R1 into bus A.
2. **SEL B** : **010** - to place the contents of R2 into bus B
3. **SEL OPR** : **10010** – to perform the arithmetic addition A+B
4. **SEL REG or SEL D : 011** – to place the result available on output bus in R3

Register and multiplexer input selection code

Binary code	SELA	SELB	SELD or SELREG
000	Input	Input	---
001	R1	R1	R1
010	R2	R2	R2
011	R3	R3	R3
100	R4	R4	R4
101	R5	R5	R5
110	R6	R6	R6
111	R7	R7	R7

Operation with symbol

Operation selection code	Operation	Symbol
0000	Transfer A	TSFA
0001	Increment A	INC A
0010	A+B	ADD
0011	A-B	SUB
0100	Decrement A	DEC
0101	A AND B	AND
0110	A OR B	OR
0111	A XOR B	XOR
1000	Complement A	COMA
1001	Shift right A	SHR
1010	Shift left A	SHL

Instruction Formats

- Computer perform task on the basis of instruction provided. A instruction in computer comprises of groups called fields.
- These field contains different information as for computers every thing is in 0 and 1 so each field has different significance on the basis of which a CPU decide what so perform.
  The most common fields are:

- Operation field which specifies the operation to be performed like addition.
- Address field which contain the location of operand, i.e., register or memory location.
- Mode field which specifies how operand is to be founded.

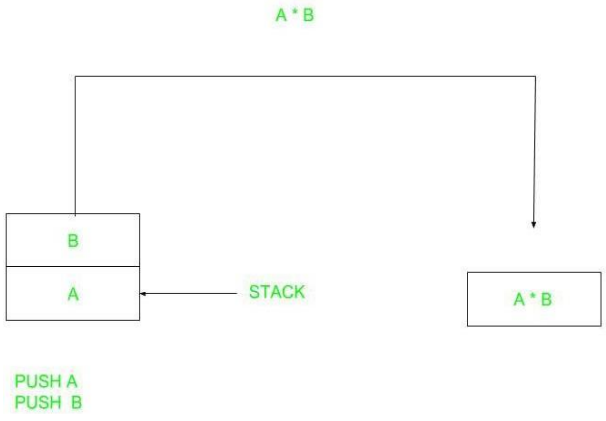
- A instruction is of various length depending upon the number of addresses it contain. Generally CPU organization are of three types on the basis of number of address fields:

1. Single Accumulator organization
2. General register organization
3. Stack organization
   - In first organization operation is done involving a special register called accumulator.
   - In second on multiple registers are used for the computation purpose. In third organization the work on stack basis operation due to which it does not contain any address field.
   - It is not necessary that only a single organization is applied a blend of various organization is mostly what we see generally.

On the basis of number of address instruction are classified as:

Note that we will use X = (A+B)*(C+D) expression to showcase the procedure.

1. **Zero Address Instructions –**



- A stack based computer do not use address field in instruction.To evaluate a expression first it is converted to revere Polish Notation i.e. Post fix Notation.

Expression: X = (A+B)*(C+D)

Postfixed : X = AB+CD+*

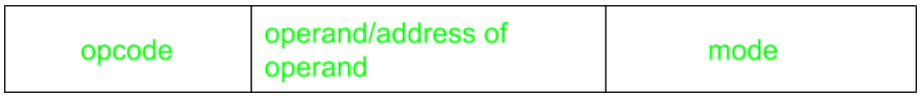
TOP means top of stack

M[X] is any memory location

PUSH	A	TOP = A
PUSH	B	TOP = B
ADD		TOP = A+B
PUSH	C	TOP = C
PUSH	D	TOP = D
ADD		TOP = C+D
MUL		TOP = (C+D)*(A+B)
POP	X	M[X] = TOP

2. One Address Instructions –

- This use a implied ACCUMULATOR register for data manipulation.
- One operand is in accumulator and other is in register or memory location.
- Implied means that the CPU already know that one operand is in accumulator so there is no need to specify it.



Expression: X = (A+B)*(C+D)

AC is accumulator

M[] is any memory location

M[T] is temporary location

LOAD	A	AC = M[A]
ADD	B	AC = AC + M[B]
STORE	T	M[T] = AC
LOAD	C	AC = M[C]
ADD	D	AC = AC + M[D]
MUL	T	AC = AC * M[T]
STORE	X	M[X] = AC

3. Two Address Instructions –

- This is common in commercial computers.Here two address can be specified in the instruction.Unlike earlier in one address instruction the result was stored in accumulator here result cab be stored at different location rather than just accumulator, but require more number of bit to represent address.

Here destination address can also contain operand.

Expression: X = (A+B)*(C+D)

R1, R2 are registers

M[] is any memory location

MOV	R1, A	R1 = M[A]
ADD	R1, B	R1 = R1 + M[B]
MOV	R2, C	R2 = C
ADD	R2, D	R2 = R2 + D
MUL	R1, R2	R1 = R1 * R2
MOV	X, R1	M[X] = R1

4. Three Address Instructions –

- This has three address field to specify a register or a memory location. Program created are much short in size but number of bits per instruction increase.
- These instructions make creation of program much easier but it does not mean that program will run much faster because now instruction only contain more information but each micro operation (changing content of register, loading address in address bus etc.) will be performed in one cycle only

Expression: X = (A+B)*(C+D)

R1, R2 are registers

M[] is any memory location

ADD	R1, A, B	R1 = M[A] + M[B]
ADD	R2, C, D	R2 = M[C] + M[D]
MUL	X, R1, R2	M[X] = R1 * R2

Adressing Modes:

- To determine the address of operand or effective address.
- The operation field of an instruction specifies the operation to be performed.
- This operation will be executed on some data which is stored in computer registers or the main memory.
- The way any operand is selected during the program execution is dependent on the addressing mode of the instruction.

Types of Addressing Modes

Below we have discussed different types of addressing modes one by one:

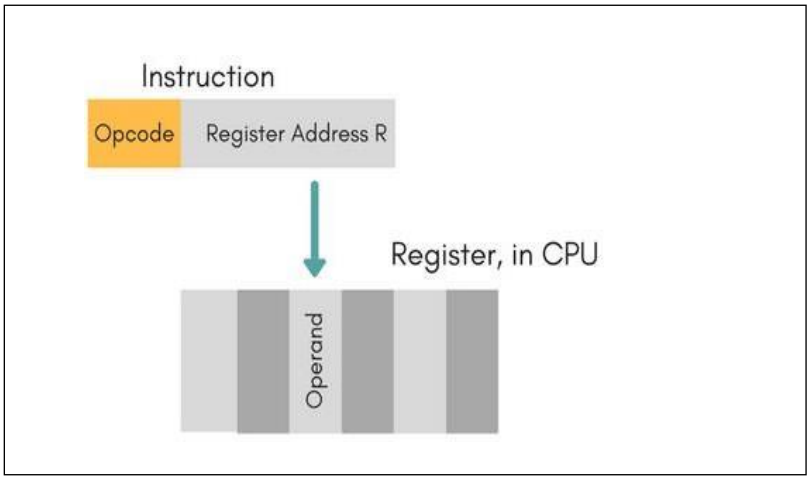
Immediate Mode:

In this mode, the operand is specified in the instruction itself. An immediate mode instruction has an operand field rather than the address field.

For example: ADD 7, which says Add 7 to contents of accumulator. 7 is the operand here.

Register Mode:

In this mode the operand is stored in the register and this register is present in CPU. The instruction has the address of the Register where the operand is stored.



Eg: Move R1,R2

Advantages

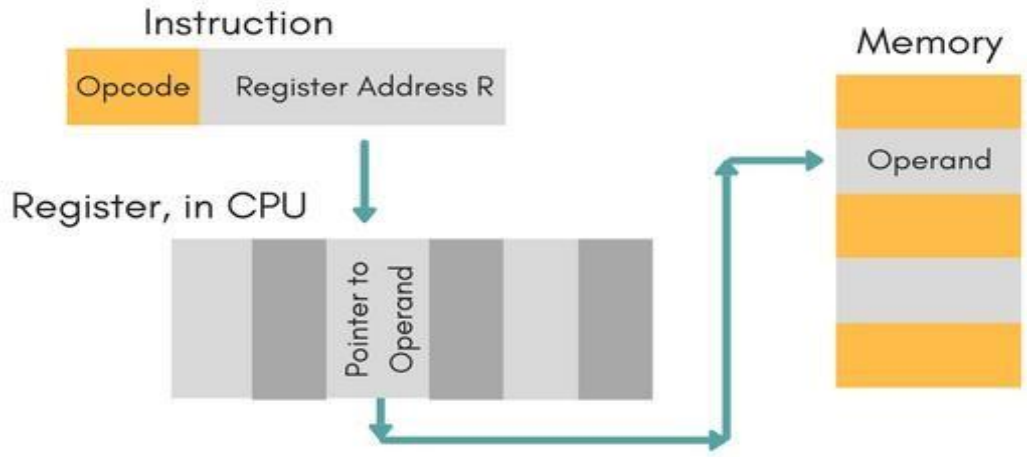
- Shorter instructions and faster instruction fetch.
- Faster memory access to the operand(s)

Disadvantages

- Very limited address space
- Using multiple registers helps performance but it complicates the instructions.

Register Indirect Mode:

In this mode, the instruction specifies the register whose contents give us the address of operand which is in memory. Thus, the register contains the address of operand rather than the operand itself.



Eg: Move A,[R0]

Auto Increment Mode:

The contents of the register is incremented to address the next location.

Eg: Move(R2)+R0

After copying the contents of R0 to R2 Register, the contents of R2 register are automatically incremented by '1'

Auto Decrement Mode:

The contents of the register specified in the instruction are decremented, then are used to access a memory location.

Eg: Move R1-(R0)

Initially decrement the contents of register R0 and then the contents of R0 are copied to R1 register.

Direct Addressing Mode:

In this mode, effective address of operand is present in instruction itself.

- Single memory reference to access data.

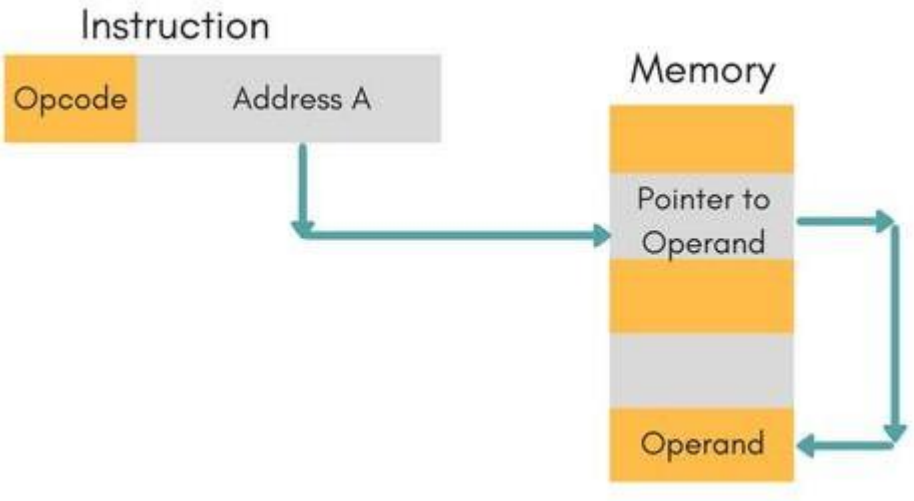
- No additional calculations to find the effective address of the operand.

For Example: ADD R1, 4000 - In this the 4000 is effective address of operand.

NOTE: Effective Address is the location where operand is present.

Indirect Addressing Mode:

In this, the address field of instruction gives the address where the effective address is stored in memory. This slows down the execution, as this includes multiple memory lookups to find the operand.



Relative Addressing Mode

In this the contents of PC(Program Counter) is added to address part of instruction to obtain the effective address.

EA = A + (PC), where EA is effective address and PC is program counter.

The operand A is address part of the instruction.

Base Register Addressing Mode

In this the contents of base register is added to the address part of the instruction to obtain the effective address

This can be defined as EA = A + (R), where A is address part of the instruction and R is base register content.

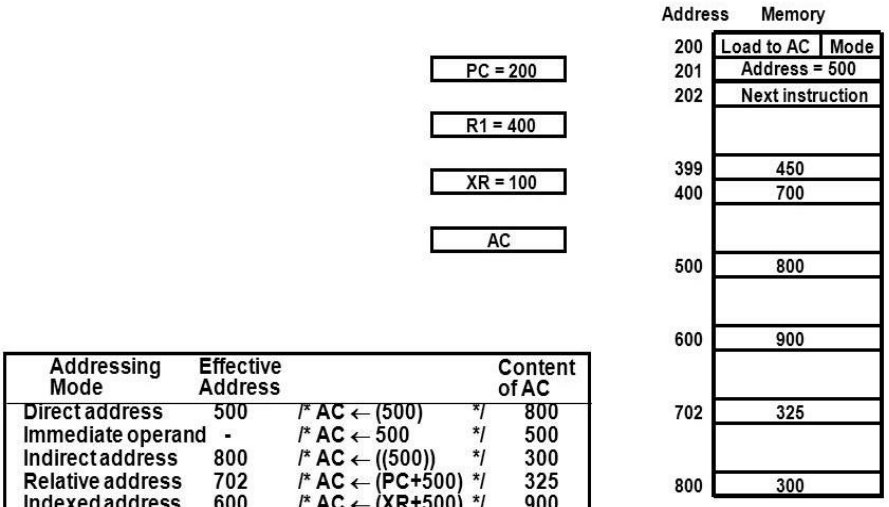
Indexed Addressing Mode

In this the contents of an index register is added to the address part of the instruction to obtain the effective address

EA = contents of Index register + (A)

A is address part of the instruction

ADDRESSING MODES - EXAMPLES



Addressing Mode	Effective Address		Content of AC
Direct address	500	/* AC ← (500) */	800
Immediate operand	-	/* AC ← 500 */	500
Indirect address	800	/* AC ← ((500)) */	300
Relative address	702	/* AC ← (PC+500) */	325
Indexed address	600	/* AC ← (XR+500) */	900
Register	-	/* AC ← R1 */	400
Register indirect	400	/* AC ← (R1) */	700
Autoincrement	400	/* AC ← (R1)+ */	700
Autodecrement	399	/* AC ← -(R1) */	450

Data Transfer Instructions:

- Data transfer instructions move data from one place in the computer to another without changing the data content.
- The most common transfers are between memory and processor registers, between processor registers and input or output, and between the processor registers themselves.
- The load instruction has been used mostly to designate a transfer from memory to a processor register, usually an accumulator.
- The store instruction designates a transfer from a processor register into memory.
- The move instruction has been used in computers with multiple CPU registers to designate a transfer from one register to another. It has also been used for data transfers between CPU registers and memory or between two memory words.
- The exchange instruction swaps information between two registers or a register and a memory word.
- The input and output instructions transfer data among processor registers and input or output terminals.

- The push and pop instructions transfer data between processor registers and a memory stack.

DATA TRANSFER INSTRUCTIONS

Typical Data Transfer Instructions

Name	Mnemonic
Load	LD
Store	ST
Move	MOV
Exchange	XCH
Input	IN
Output	OUT
Push	PUSH
Pop	POP

Data Transfer Instructions with Different Addressing Modes

Mode	Assembly Convention	Register Transfer
Direct address	LD ADR	AC ← M[ADR]
Indirect address	LD @ADR	AC ← M[M[ADR]]
Relative address	LD $ADR	AC ← M[PC + ADR]
Immediate operand	LD #NBR	AC ← NBR
Index addressing	LD ADR(X)	AC ← M[ADR + XR]
Register	LD R1	AC ← R1
Register indirect	LD (R1)	AC ← M[R1]
Autoincrement	LD (R1)+	AC ← M[R1], R1 ← R1 + 1
Autodecrement	LD -(R1)	R1 ← R1 - 1, AC ← M[R1]

DATA MANIPULATION INSTRUCTIONS:

DATA MANIPULATION INSTRUCTIONS

Three Basic Types: **Arithmetic instructions** **Logical and bit manipulation instructions**

Shift instructions

Arithmetic Instructions

Name	Mnemonic
Increment	INC
Decrement	DEC
Add	ADD
Subtract	SUB
Multiply	MUL
Divide	DIV
Add with Carry	ADDC
Subtract with Borrow	SUBB
Negate(2's Complement)	NEG

Logical and Bit Manipulation Instructions

Name	Mnemonic
Clear	CLR
Complement	COM
AND	AND
	OR
OR	XOR
	CLRC
Exclusive-OR	SETC
Clear carry	
Set carry	
Complement carry	COMC
Enable interrupt	EI
Disable interrupt	DI

Shift Instructions

Name	Mnemonic
Logical shift right	SHR
Logical shift left	SHL
Arithmetic shift right	SHRA
Arithmetic shift left	SHLA
Rotate right	ROR
Rotate left	ROL
Rotate right thru carry	RORC
Rotate left thru carry	ROLC

PROGRAM CONTROL INSTRUCTIONS:

- It is sometimes convenient to supplement the ALU circuit in the CPU with a statusregister where status bit conditions be stored for further analysis. Status bits are also called condition-code bits or flag bits.
- Figure 5.3 shows the block diagram of an 8-bit ALU with a 4-bit status register. The four status bits are symbolized by C, S, Z, and V. The bits are set or cleared as a result of an operation performed in the ALU.

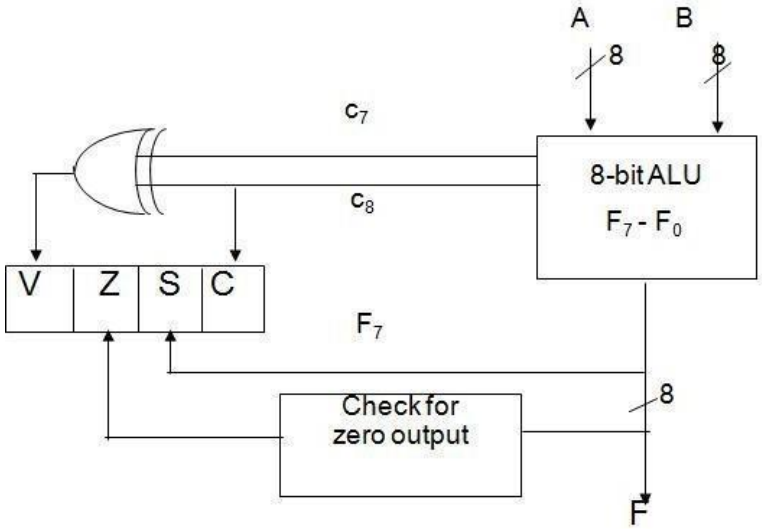


Figure 5.3: Status Register Bits

1. Bit C (carry) is set to 1 if the end carry C8 is 1. It is cleared to 0 if the carry is 0.
2. Bit S (sign) is set to 1 if the highest-order bit F7 is 1. It is set to 0 if set to 0 if the bit is 0.
3. Bit Z (zero) is set to 1 if the output of the ALU contains all 0's. it is cleared to 0 otherwise. In other words, Z = 1 if the output is zero and Z = 0 if the output is not zero.
4. Bit V (overflow) is set to 1 if the exclusives-OR of the last two carries is equal to 1, and cleared to 0 otherwise. This is the condition for an overflow when negative numbers are in 2's complement. For the 8-bit ALU, V = 1 if the output is greater than + 127 or less than -128.

- The status bits can be checked after an ALU operation to determine certain relationships that exist between the vales of A and B.
- If bit V is set after the addition of two signed numbers, it indicates an overflow condition.
- If Z is set after an exclusive-OR operation, it indicates that A = B.
- A single bit in A can be checked to determine if it is 0 or 1 by masking all bits except thebitin question and then checking the Z status bit.

Name	Mnemonic
Branch	BR
Jump	JMP
Skip	SKP
Call	CALL
Return	RTN
Compare(by -)	CMP
Test (by AND)	TST

CMP and TST instructions do not retain their results of operations (- and AND, respectively). They only set or clear certain Flags.