InstructionSetArchitecture

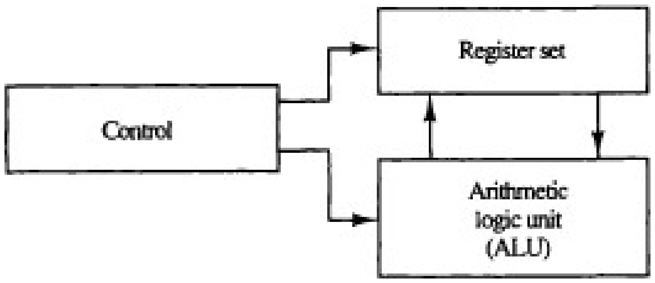
Central processing unit:

INTRODUCTION

The part of the computer that performs the bulk of data-processing operations is called thecentralprocessingunitandisreferredtoastheCPU.TheCPUismadeupofthreemajorparts,as shown in Fig. The register set stores intermediate data used during the execution of the instructions. The arithmetic logic unit (ALU) performs the required micro operations forexecutingtheinstructions.Thecontrolunitsupervisesthetransferofinformationamongtheregister sandinstructsthe ALU as to which operation to perform.

The CPU performs a variety of functions dictated by the type of instructions that are incorporated in the computer. Computer architecture is sometimes defined as the computer structure and behavior as seen by the programmer that uses machine language instructions. This includes the instruction formats, addressing modes, the instruction set, and the general organization of the CPU registers.

Components of CPU



General register organization

A bus organization for seven CPU registers is shown in Fig. The output of each register is connected to two multiplexers (MUX) to form the two buses A and B . The selection lines in each multiplexer select one register or the input data for the particular bus.

The A and B buses form the inputs to a common arithmetic logic unit (ALU). The operations elected in the ALU determines the arithmetic or logic micro operation that is to be performed. The result of the micro operation is available for output data and also goes into the inputs of all the registers. The register that receives the information from the output bus is selected by a decoder. The decoder activates one of the register load inputs, thus providing a transfer path between the data in the output bus and the inputs of the selected destination register.

The control unit that operates the CPU bus system directs the information flow through theregistersandALUbyselectingthevariouscomponentsinthesystem.Forexample,to perform

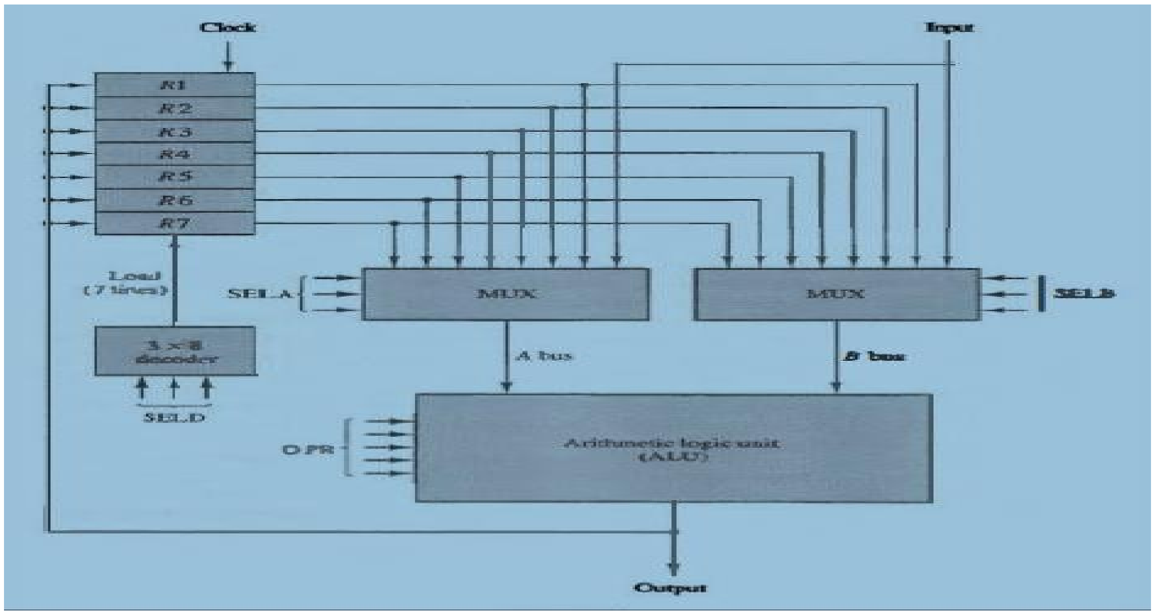
the operation R 1 <--R2 + R3 the control must provide binary selection variables to the following selector inputs:

1. MUX A selector (SELA): to place the content of R2 into bus A .
2. MUX B selector (SELB): to place the content o f R 3 into bus B .
3. ALU operation selector (OPR): to provide the arithmetic addition A + B .
4. Decoder destination selector (SELD): to transfer the content of the output bus into R1

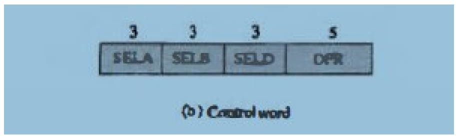
.

The four control selection variables are generated in the control unit and must be available at the beginning of a clock cycle. The data from the two source registers propagate through the gates in the multiplexers and the ALU, to the output bus, and into the inputs of the destination register, all during the clock cycle interval. Then, when the next clock transition occurs, the binary information from the output bus is transferred into R 1.

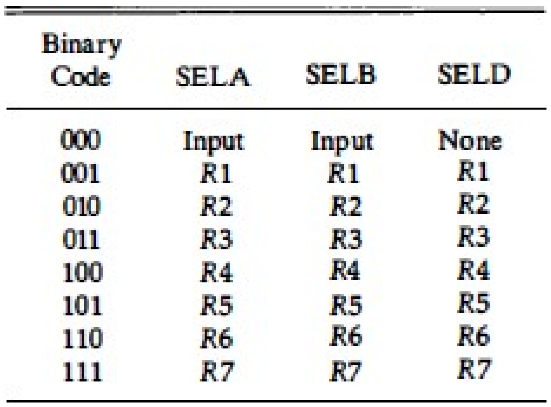
To achieve a fast response time, the ALU is constructed with high speed circuits.



Control Word



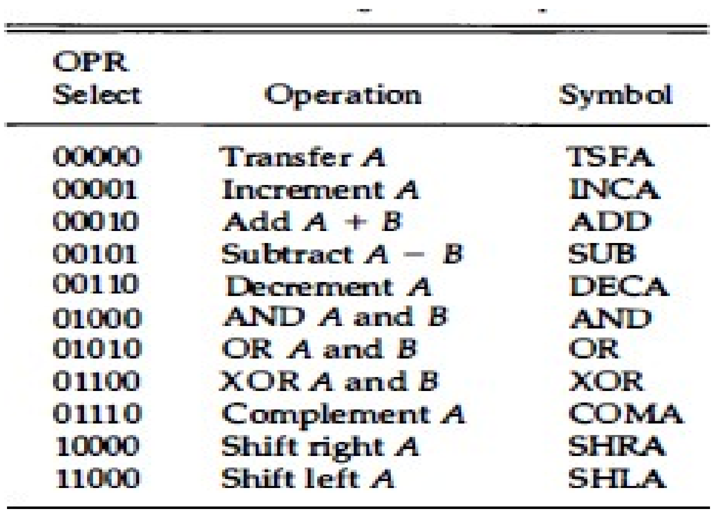
There are 14 binary selection inputs in the unit, and their combined value specifies a control word .Encoding of registers election fields



Binary Code	SELA	SELB	SELD
000	Input	Input	None
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

ALU

The ALU provides arithmetic and logic operations. In addition, the CPU must provide shiftoperations.TheshiftermaybeplacedintheinputoftheALUtoprovideapreshiftcapability,or at the output of the ALU to provide post shifting capability. In some cases, the shift operations are included with the ALU.

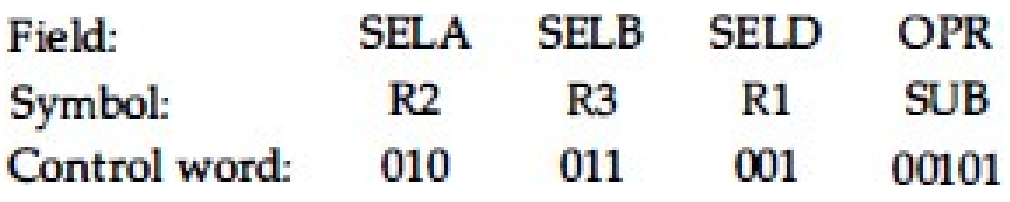


OPR Select	Operation	Symbol
00000	Transfer *A*	TSFA
00001	Increment *A*	INCA
00010	Add *A* + *B*	ADD
00101	Subtract *A* − *B*	SUB
00110	Decrement *A*	DECA
01000	AND *A* and *B*	AND
01010	OR *A* and *B*	OR
01100	XOR *A* and *B*	XOR
01110	Complement *A*	COMA
10000	Shift right *A*	SHRA
11000	Shift left *A*	SHLA

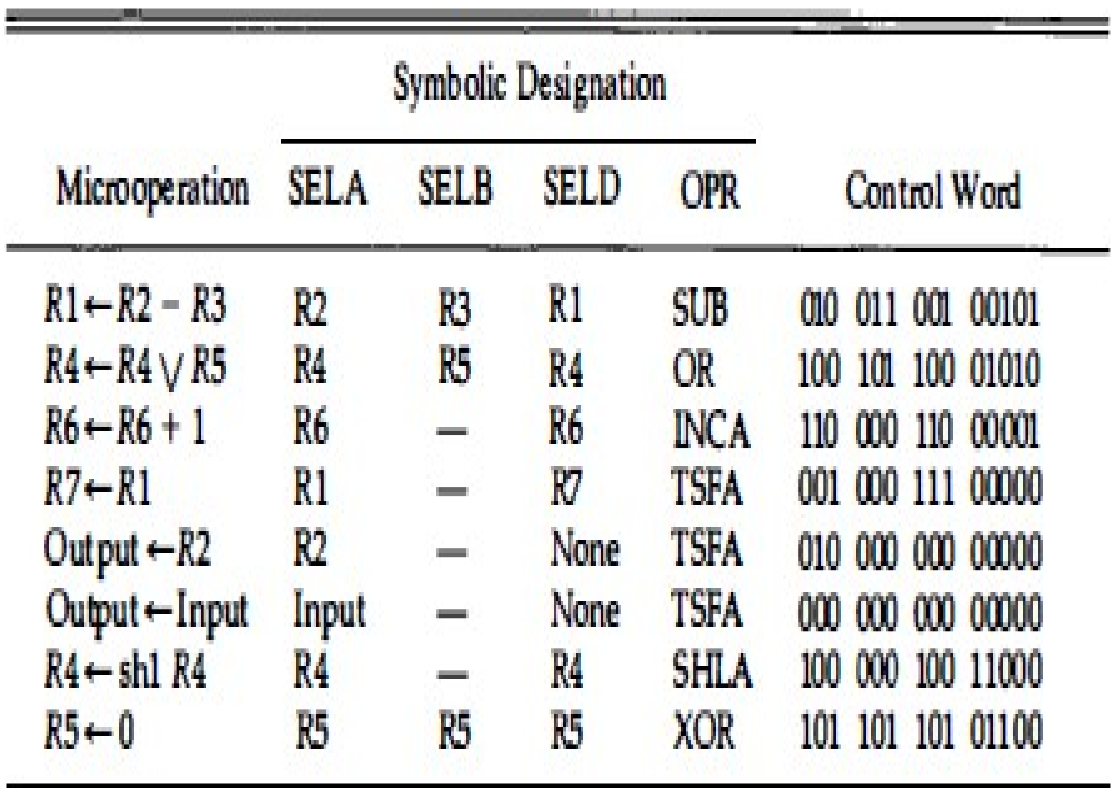
Examples of Micro operations R

1 <- R 2 - R3

specifies R2 for the A input of the ALU, R3 for the B input of the ALU, R1 for the destination register, and an ALU operation to subtract A -B.



Field:	SELA	SELB	SELD	OPR
Symbol:	R2	R3	R1	SUB
Control word:	010	011	001	00101



	Symbolic Designation				
Microoperation	SELA	SELB	SELD	OPR	Control Word
R1←R2 − R3	R2	R3	R1	SUB	010 011 001 00101
R4←R4 ∨ R5	R4	R5	R4	OR	100 101 100 01010
R6←R6 + 1	R6	—	R6	INCA	110 000 110 00001
R7←R1	R1	—	R7	TSFA	001 000 111 00000
Output←R2	R2	—	None	TSFA	010 000 000 00000
Output←Input	Input	—	None	TSFA	000 000 000 00000
R4←shl R4	R4	—	R4	SHLA	100 000 100 11000
R5←0	R5	R5	R5	XOR	101 101 101 01100

InstructionFormats

The most common fields found in instruction formats are :

1. An operation code field that specifies the operation to be performed.
2. An address field that designates a memory address or a process or register.
3. A mode field that specifies the way the operand or the effective address is determined.

Most computers fall in to one of three types of CPU organizations:

1. Single accumulator organization.
2. General register organization.
3. Stack organization

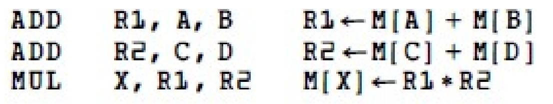
To illustrate the influence of the number of addresses on computer programs, we will evaluate the arithmetic statement

X=(A+B)•(C+D)

Using zero, one, two, or three address instructions.

Three-Address Instructions

Computerswiththree-addressinstructionformatscanuseeachaddressfieldtospecifyeitheraprocessorregisteroramemoryoperand.TheprograminassemblylanguagethatevaluatesX=(A+B)•(C+D)isshownbelow,togetherwithcommentsthatexplaintheregistertransferoperationofeachinstruction.



```
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
```

It is assumed that the computer has two processor registers, R 1 and R2. The symbol M [A]denotes the operand at memory address symbolized by A . The advantage o f the three-address format i s that i t results in short programs when evaluating arithmetic expressions. The disadvantage is that the binary-coded instructions require too many bits to specify three addresses.

An example of a commercial computer that uses three-address instructions is the Cyber 170.The instruction formats in the Cyber computer are restricted to either three register address fields or two register address fields and one memory address field.

Two-Address Instructions

Two-address instructions are the most common in commercial computers. Here again each address field can specify either a processor register or a memory word. The program to evaluate $X = (A + B) * (C + D)$ is as follows:

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

The MOV instruction moves or transfers the operands to and from memory and processor registers. The first symbol listed in an instruction is assumed to be both a source and the destination where the result of the operation is transferred.

One-Address Instructions

One-address instructions use an implied accumulator (AC) register for all data manipulation. For multiplication and division there is a need for a second register. However, here we will neglect the second register and assume that the AC contains the result of all operations. The program to evaluate $X = (A + B) * (C + D)$ is

```
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
```

All operations are done between the AC register and a memory operand. T is the address of a temporary memory location required for storing the intermediate result.

Zero-Address Instructions

A stack-organized computer does not use an address field for the instructions ADD and MUL. The PUSH and POP instructions, however, need an address field to specify the operand that communicates with the stack. The following program shows how $X = (A + B) * (C + D)$ will be written for a stack-organized computer. (TOS stands for top of stack.)

```
PUSH   A    TOS ← A
PUSH   B    TOS ← B
ADD         TOS ← (A + B)
PUSH   C    TOS ← C
PUSH   D    TOS ← D
ADD         TOS ← (C + D)
MUL         TOS ← (C + D) * (A + B)
POP    X    M[X] ← TOS
```

To evaluate arithmetic expressions in a stack computer, it is necessary to convert the expression into reverse Polish notation. The name "zero-address" is given to this type of computer because of the absence of an address field in the computational instructions.

Addressing Modes

3. Addressing Modes :

The operation field of an instruction specifies the operation to be performed. This operation must be executed on some data stored in computer registers or memory words. The way the operands are chosen during program execution independent on the addressing mode of the instruction. The addressing mode specifies a rule for interpreting or modifying theaddressfieldoftheinstructionbeforetheoperandisactuallyreferenced.
Computers use addressing mode techniques for the purpose of accommodating one or both of the following provisions:

1 To give programming versatility to the user by providing such facilities as pointers to Memory, counters for loop control, indexing of data ,and program relocation
2 To reduce the number of bits in the addressing field of the instruction.
3 The availability of the addressing modes gives the experienced assembly language programmer flexibility for writing programs that are more efficient with respect to the number of instructions and execution time.

To understand the various addressing modes to be presented in this section ,It is imperative that we understand the basic operation cycle of the computer. The control unit of a computer is designed to go through an instruction cycle that is divided into three major phases:

1. Fetch the instruction from memory
2. Decode the instruction.
3. Execute the instruction.

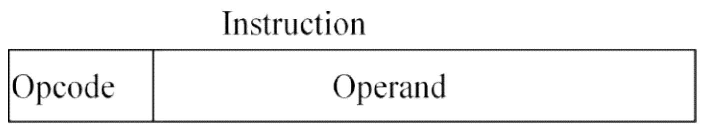
Addressing modes are as:

1. Implied Mode

Address of the operands are specified implicitly in the definition of the instruction
- No need to specify address in the instruction
- EA=AC, or EA=Stack[SP**],** **EA : Effective Address.**

2. Immediate Mode

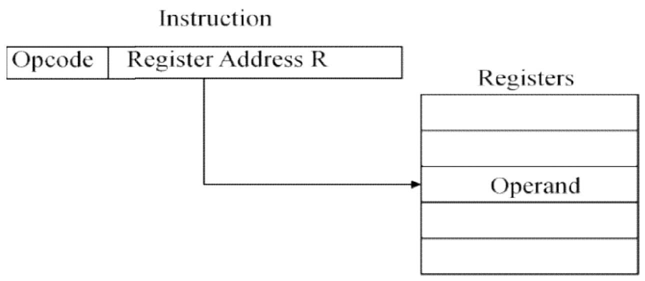
Instead of specifying the address of the operand ,operand itself is specified
- No need to specify address in the instruction
- However, operand itself needs to be specified
- Sometimes, require more bits than the address
- Fast to acquire an operand



EA=Not defined.

3. Register Mode

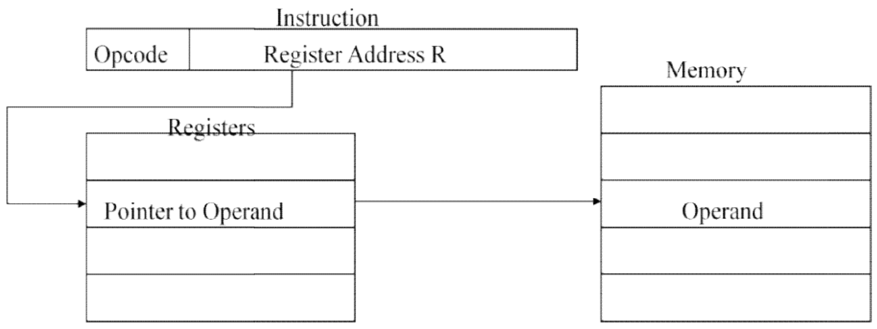
Address specified in the instruction is the register address.
- Designated operand need to be in a register
- Shorter address than the memory address
- Saving address field in the instruction
- FastertoacquireanoperandthanthememoryaddressingEA= IR(R)(IR(R):RegisterfieldofIR)



4. Register Indirect Mode

Instruction specifies a register which contains the memory address of the operand

- Savinginstructionbitssinceregisteraddressiss horterthanthememoryaddress
- Slowertoacquireanoperandthanboththeregi steraddressingormemoryaddressing
- EA=[IR(R)]([x]: Content of x)



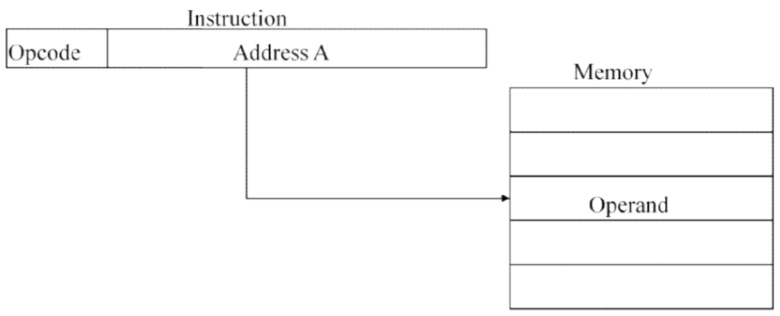
5.Auto-incrementor Auto-decrement features:

Same as the Register Indirect, but, when the address in the register is used to access memory, the value in the register is incremented or decremented by 1 (after or before the execution of the instruction).

6. Direct Address Mode

Instruction specifies the memory address which can be used directly to the physical memory

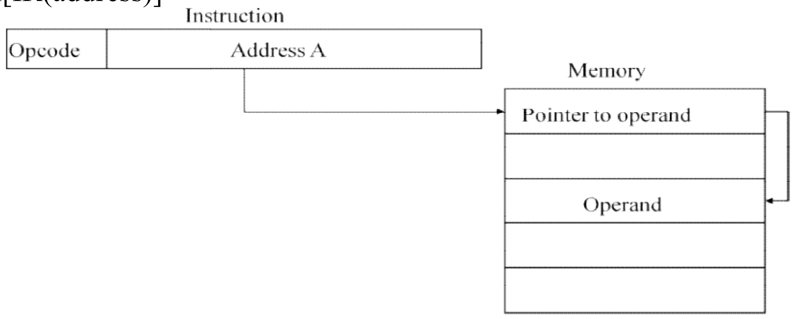
- Faster than the other memory addressing modes
- Too many bits are needed to specify the address for a large physical memory space
- EA=IR(address),(IR(address): address field of IR)



7.Indirect Addressing Mode

Theaddressfieldofaninstructionspecifiestheaddressofamemorylocationthatcontainstheaddress of the operand

- Whentheabbreviatedaddressisused,largephysicalmemorycanbeaddressedwitharelativel ysmallnumberofbits
- Slow to acquire an operand because of an additional memory access
- EA=M[IR(address)]



7. Displacement Addressing Mode

The Address fields of an instruction specifies the part of the address (abbreviated address) which can be used along with a designated register to calculate the address of the operand

a) PC Relative Addressing Mode(R=PC)

- EA= PC+IR(address)
- Address field of the instruction is short
- Large physical memory can be accessed with a small number of address bits

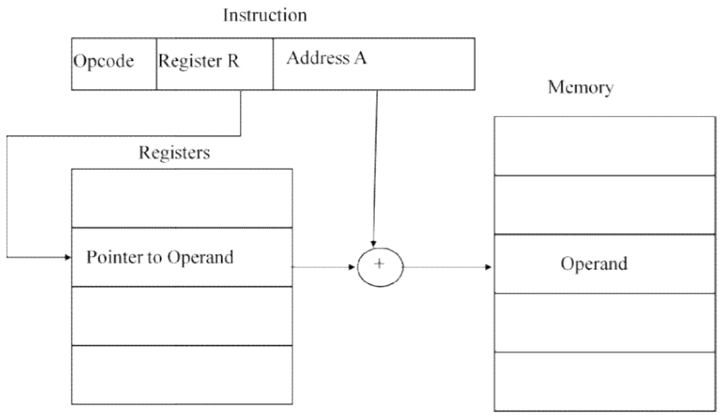
b) Indexed Addressing Mode

- XR: Index Register:
- EA= XR+ IR(address)

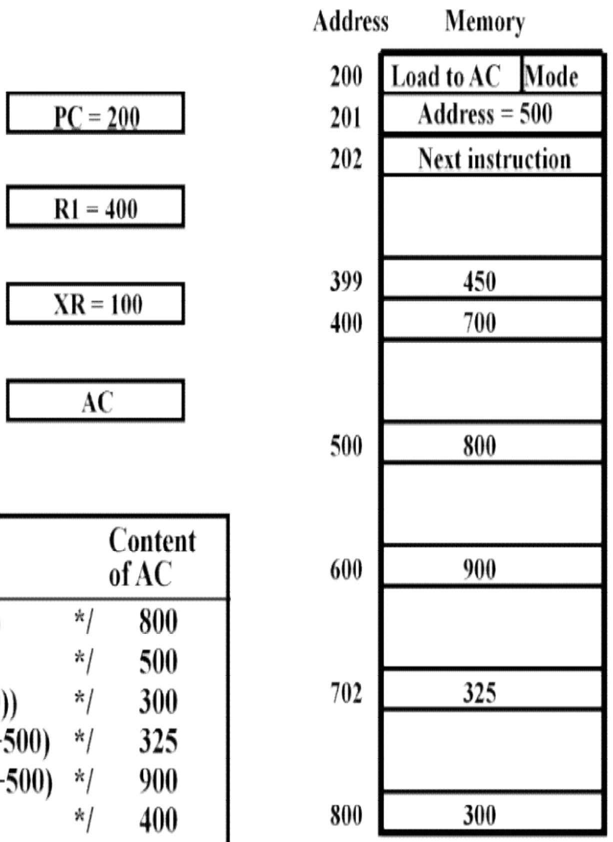
c) Base Register Addressing Mode

BAR: Base Address Register:

- EA=BAR+IR(address)



Numerical Example:



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 ← -(R) */	450

Data Transfer & Manipulation

Computer provides an extensive set of instructions to give the user the flexibility to carryout various computational tasks. Most computer instruction can be classified into three categories.

(1) Data transfer instruction
(2) Data manipulation instruction
(3) Program control instruction

Data transfer instruction cause transferred data from one location to another without changingthebinaryinstructioncontent.Datamanipulationinstructionsarethosethatperformarithmeticlogic, and shift operations. Program control instructions provide decision-making capabilitiesandchangethepathtakenbytheprogramwhenexecutedinthecomputer.

(1) Data Transfer Instruction

Data transfer instruction move data from one place in the computer to another withoutchangingthedatacontent.Themostcommontransfersarebetweenmemoryandprocessesregisters, between processes register & input or output, and between processes register themselves

(Typical data transfer instruction)

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

(2) Data Manipulation Instruction

It performs operations on data and provides the computational capabilities for the computer. The data manipulation instructions in a typical computer are usually divided into three basic types.

(a) Arithmetic Instruction
(b) Logical bit manipulation Instruction
(c) Shift Instruction.

(a) Arithmetic Instruction

Name	**Mnemonic**
Increment	INC
Decrement	DEC
Add	Add
Subtract	Sub
Multiply	MUL
Divide	DIV
Add with Carry	ADDC
Subtract with Basses	SUBB
Negate(2'sComplement)	NEG

(b) Logical & Bit Manipulation Instruction

Name	Mnemonic
Clear	CLR
Complement	COM
AND	AND
OR	OR
Exclusive-Or	XOR
Clear Carry	CLRC
Set Carry	SETC
Complement Carry	COMC
Enable Interrupt	ET
Disable Interrupt	OI

(c) Shift Instruction

Instructions to shift the content of an operand are quite useful and one often provided in several variations. Shifts are operation in which the bits of a word are moved to the left or right. The bit-shifted in at the end of the word determines the type of shift used. Shift instruction may specify either logical shift, arithmetic shifts, or rotate type shifts.

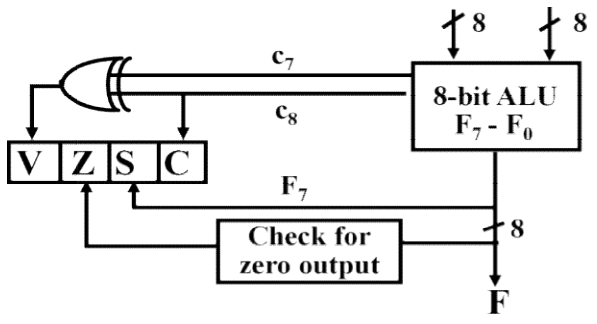
Name	Mnemonic
Logical Shif tright	SHR
Logical Shift left	SHL
Arithmetic shift right	SHRA
Arithmetic shift left	SHLA
Rotate right	ROR
Rotate left	ROL
Rotate right through carry	RORC
Rotate left through carry	ROLC

(3)Program Control:

Instructions are always stored in successive memory locations. When processed in theCPU,the instructions are fetched from consecutive memory locations and executed

Status Bit Conditions

It is sometimes convenient to supplement the ALU circuit in the CPU with a status register where status b it conditions can be stored for further analysis. Status bits are also called condition-code



bits or flag bits. 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.

1. BitC(carry)issetto1iftheendcanyC8is1.Itisclearedto0ifthecarryis0.
2. BitS(sign)issetto1ifthehighest-orderbitF?is1. Itissetto0ifthebitis0.
3. BitZ(zero)issetto1iftheoutputoftheALUcontainsall0's.Itisclearedto0otherwise.

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 exclusive-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.