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

Instruction

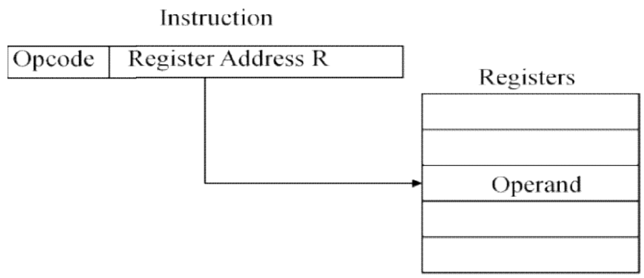
Opcode	Operand

EA=Not defined.

3. Register Mode

Address specified in the instruction is the register address.

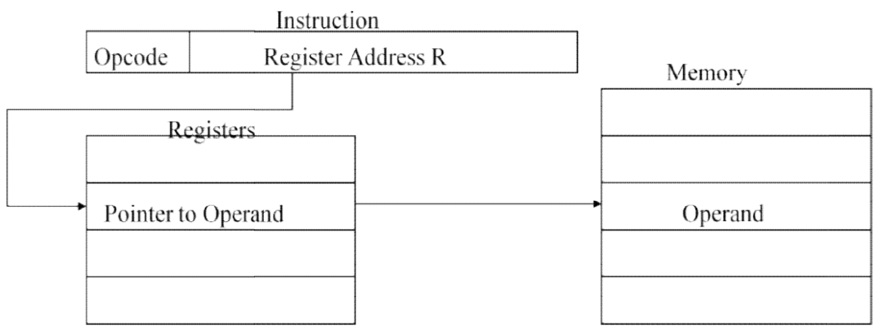
- Designated operand need to be in a register
- Shorter address than the memory address
- Saving address field in the instruction
- FastertoacquireanoperandthanthememoryaddressingEA= IR(R)(IR(R):RegisterfieldofIR)



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)



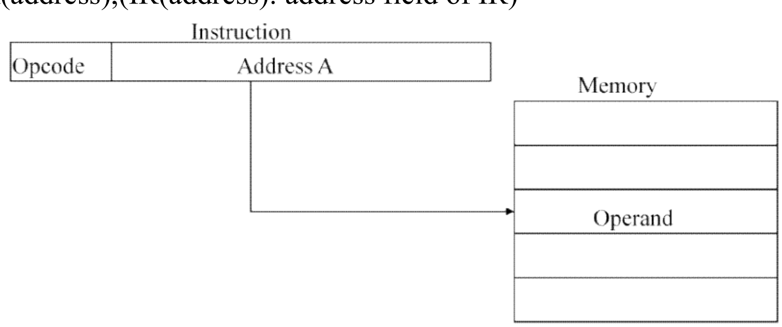
5.Auto-incrementor Auto-decrement features:

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

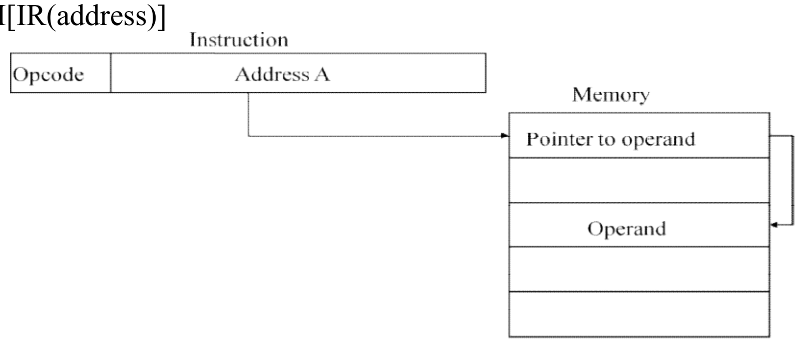
- Faster than the other memory addressing modes
- Too many bits are needed to specify the address for a large physical memory space
- EA=IR(address),(IR(address): address field of IR)



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

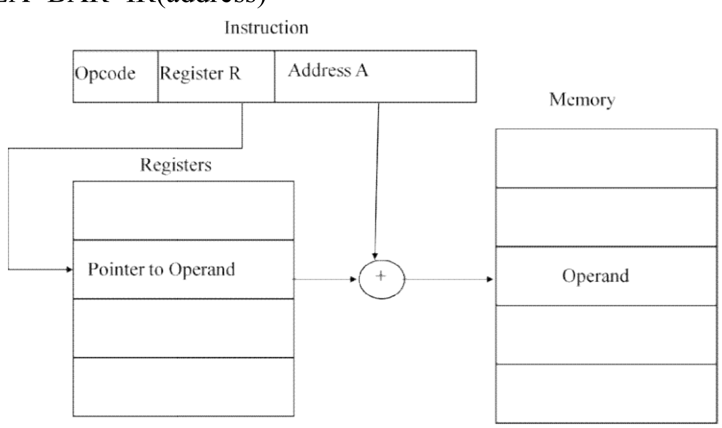
b) Indexed Addressing Mode

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

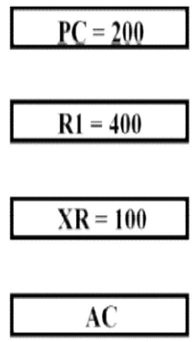
c) Base Register Addressing Mode

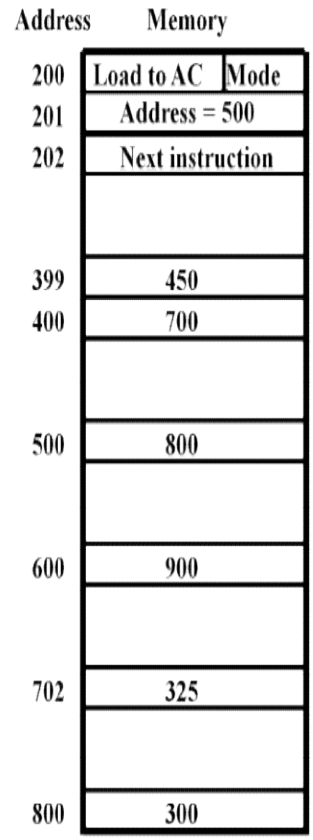
BAR: Base Address Register:

- 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.Datamanipulationinstructionsarethosethatperformarithmeti clogic, 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.Themostcommontransfersarebetweenmemoryandprocessesr egisters, 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.

<u>(a) Arithmetic Instruction</u>

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.

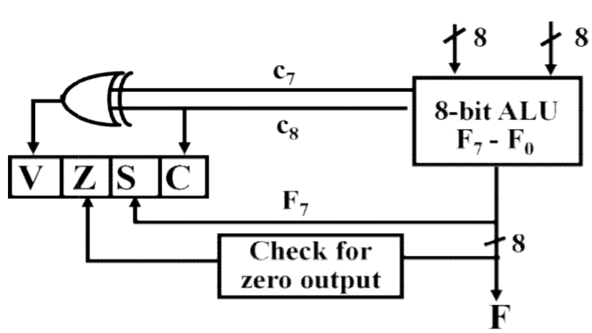
Name	Mnemonic
Logical Shif tright	SHR
Logical Shift left	SHL
Arithmetic shift right	SHRA
Arithmetic shift left	SHLA
Rotate right	ROR
Rotate left	ROL
Rotate right through carry	RORC
Rotate left through carry	ROLC

(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.