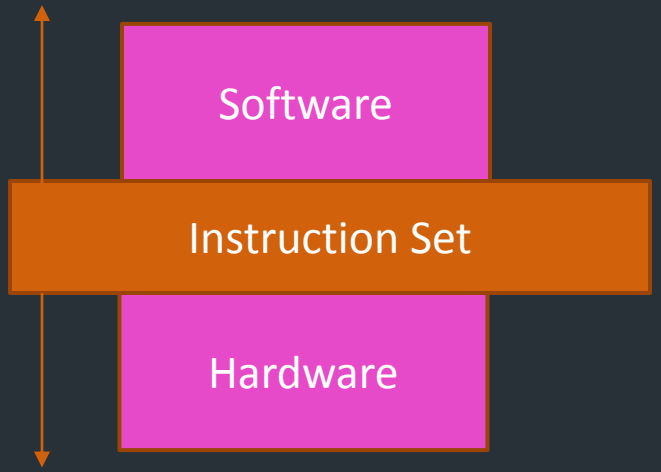
MODULE 3
Fundamentals of computer architecture
PART 1A

Dr. B. Bhanu Chander
SCOPE,
VIT University, Chennai campus

Fundamentals of Computer Architecture

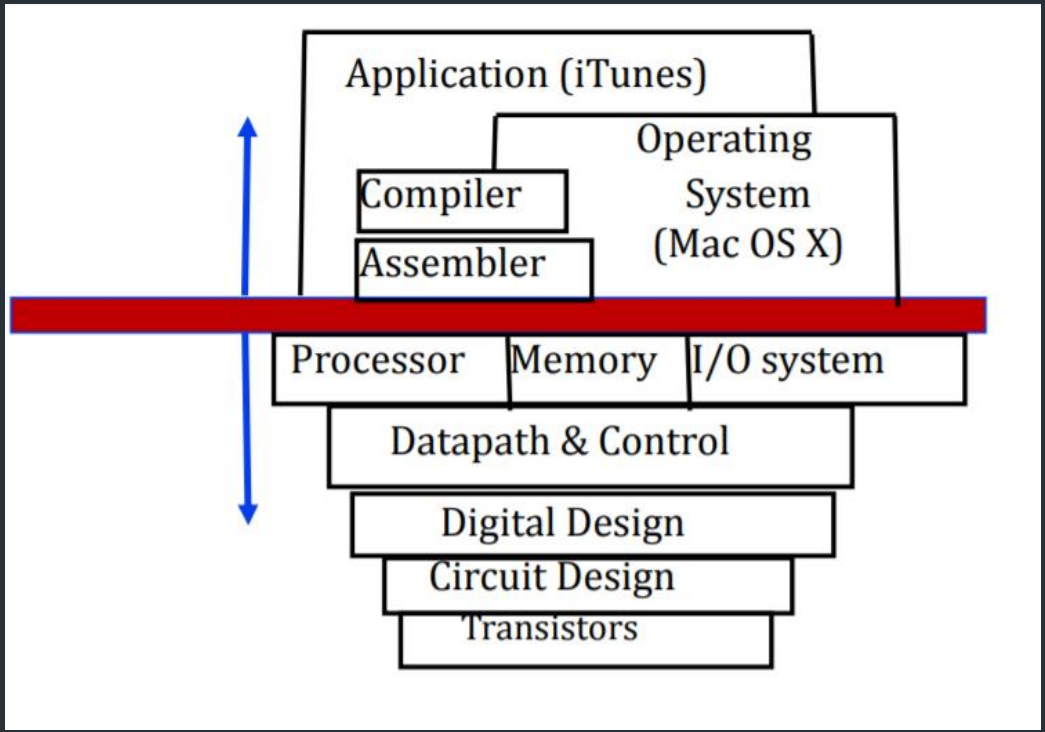
- ISA
- Instruction Formats
- Instruction types
- Addressing Modes
- Instruction execution (phases of instruction cycle)
- Assembly language programming
- Subroutine call and return mechanisms

Instruction Set Architecture



- Interface between hardware and software
- Interface between high level language and machine level language

ISA…


Application (iTunes)
Operating System (Mac OS X)
Compiler
Assembler
Processor
Memory
I/O system
Datapath & Control
Digital Design
Circuit Design
Transistors

Elements of an ISA…

- Instructions
  - Instruction Formats
  - Addressing Modes
  - Condition Codes
  - Instruction sets

Characteristics of good ISA

- Must be Clear
- Less usage of complex instructions
- Ease of compilation
- Ease of implementation

INSTRUCTION FORMATS

Instruction Formats

- 3-Address Instructions
  - ADD op1, op2, op3 ;op1 ← op2 + op3
- 2- Address Instructions
  - ADD op1,op2 ;op1 ← op1+op2

Instruction Formats…

- 1-Address Instructions
  - INC op1 ;op1 ←op1+1
- 0-Address Instructions
  - PUSH A TOS ← A

Instruction Formats…

.

Estimate this expression using all the instruction formats

$$(A + B) \times (C + D)$$

Addressing Mode

- 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.

Addressing Mode

Effective Address (EA):

Effective address is the **address of the exact memory location** where the **value of the operand** is present.

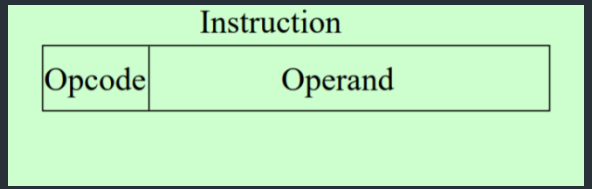
Types of addressing mode

- Implied mode
- Immediate mode
- Register mode
- Register indirect mode
- Auto increment or auto decrement mode
- Direct address mode
- Indirect address mode
- Relative address mode
- Indexed address mode
- Base register mode

Source: **Computer Organization by** HAMACHER

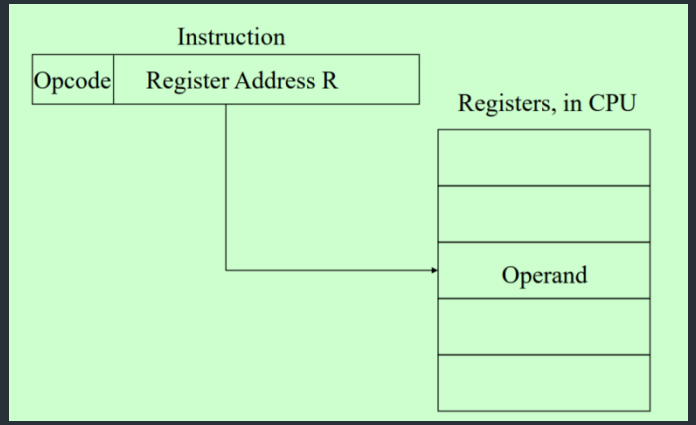
Addressing Modes…

- Implied
  - AC is implied in “ADD M[AR]” in “One-Address” instr.
  - TOS is implied in “ADD” in “Zero-Address” instr.
- Immediate
  - The use of a constant in “MOV R1, 5”, i.e. R1 ← 5
  - Operand is part of instruction
  - Operand = operand field
  - e.g. ADD 5 —Add 5 to contents of accumulator —5 is the operand

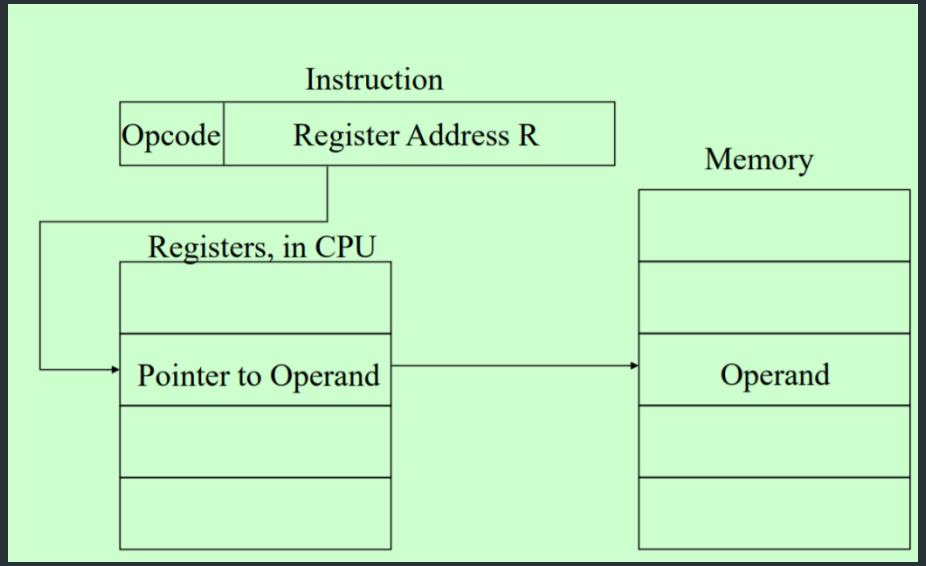


Register Mode

- Register
  - Indicate which register holds the operand
  - Operand is held in register named in address filed
  - Effective Address = R



Register Indirect addressing mode



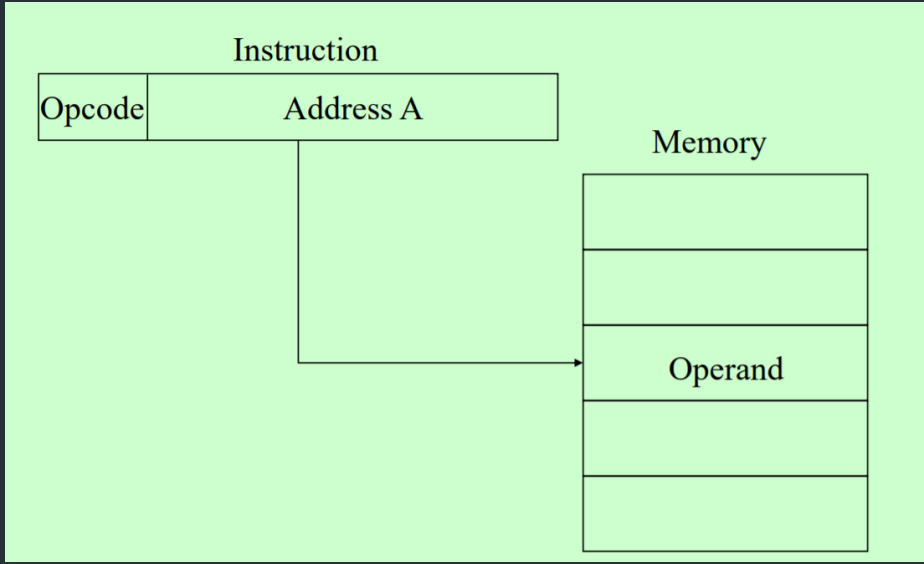
Autoincrement/ Autodecrement

- Autoincrement / Autodecrement
  - It gets incremented or decremented by 1.

Direct Addressing Mode

- Direct addressing mode ADD R1,R2

Use the given address to access a memory location

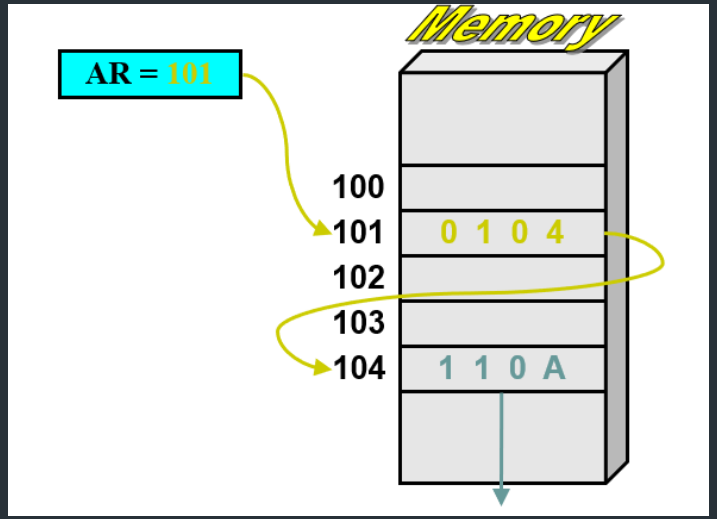


Direct Addressing Mode..

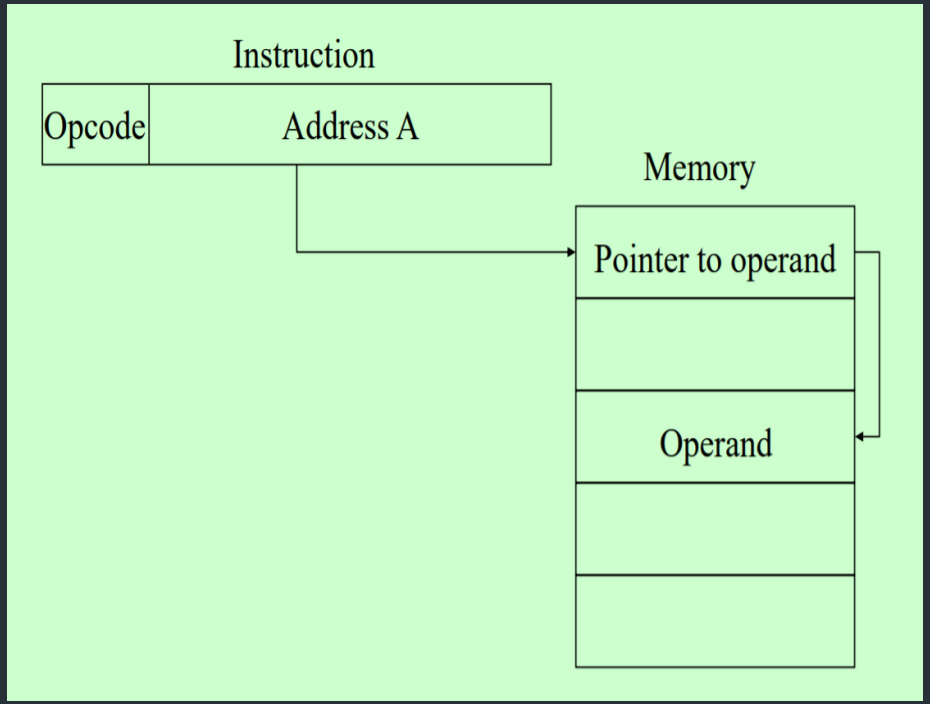
- Address field contains address of operand
- Effective address (EA) = address field (A)
- e.g. ADD A
  - —Add contents of cell A to accumulator
  - —Look in memory at address A for operand

Indirect Addressing Mode

- Indirect Address
  - Indicate the memory location that holds the address of the memory location that holds the data

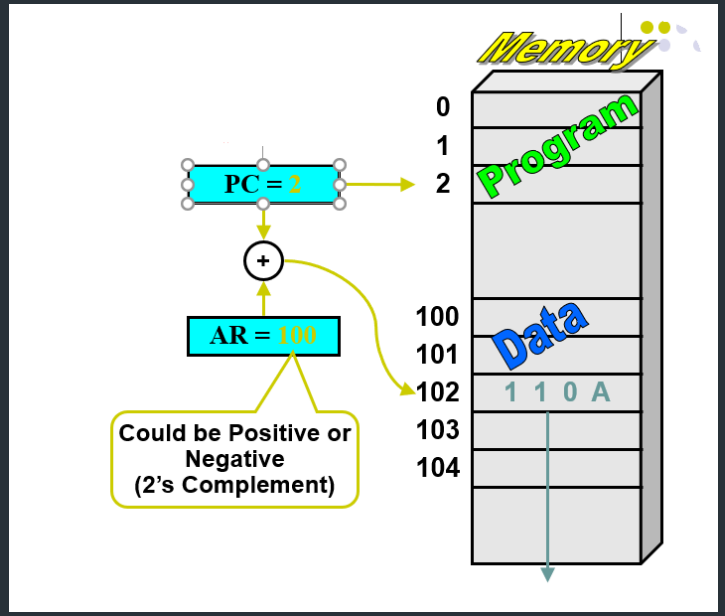


Indirect Addressing Mode



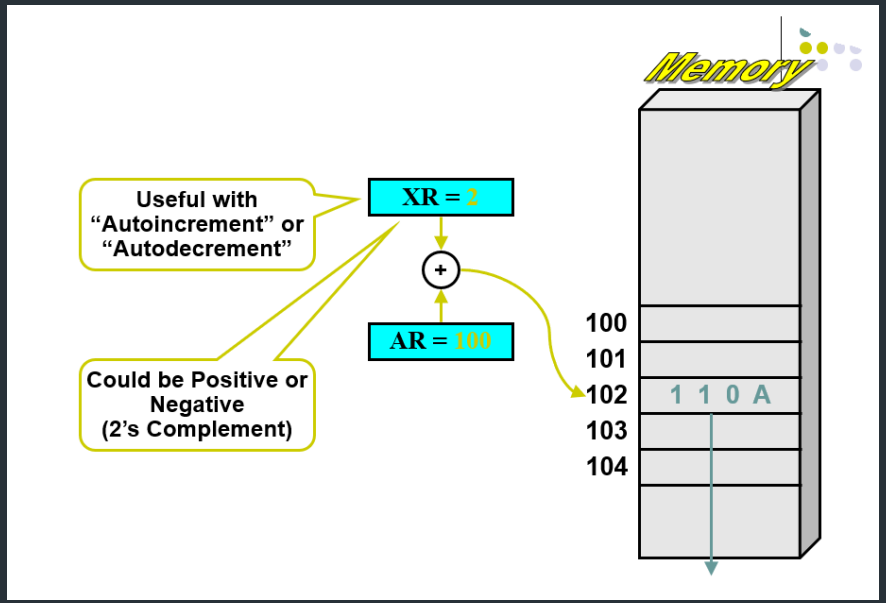
Relative Mode

- Relative Address
  - *EA* = PC + Relative Addr



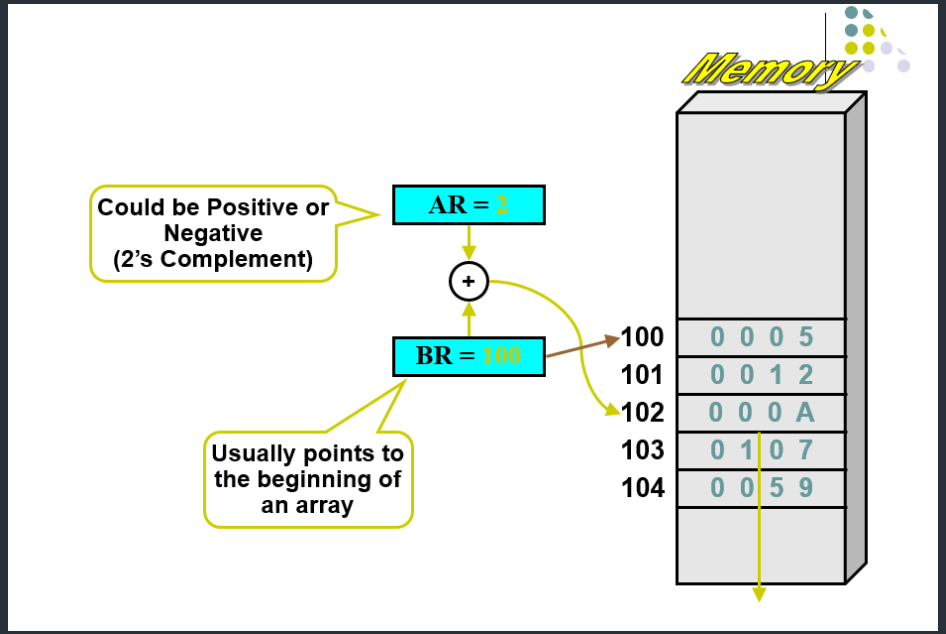
Indexed Mode…

- Indexed
  - *EA* = Index Register + Relative Addr



Base Register mode…

- BR
  - *EA* = BR + Relative Addr



Assembly Language Notation

Assembly Language

- It is the format to represent the machine instructions and programs.
- MOVE LOC, R1 ;it transfers from memory location LOC to R1
- ADD R1, R2, R3 ;Add two numbers placed in R1 and R2;
  Sum is placed in R3

INSTRUCTION TYPES

Instruction Types

- Data Transfer Instructions

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

Data Manipulation instructions

Data manipulation instructions perform operations on data and provide the computational capabilities for the computer. The data manipulation instruction in a computer are usually divided into three basic types: 1. Arithmetic instructions 2. Logical and Bit manipulation instructions 3. Shift instructions.

Instruction Types…

- Data manipulation 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	NEG

Arithmetic Instructions:

- The four basic arithmetic operations are addition, subtraction, multiplication & division. Some computers have only addition &subtraction instruction .
- Increment (INC):-this instruction adds 1 to the value stored in a register or memory word. - One common characteristic of increment operations when executed in processor registers is that a binary number of all 1’s when incremented produces a result of all 0’s.
- Decrement (DEC):-this instruction subtracts 1 from the value stored in a register or memory word. - One common characteristic of decrement operations when executed in processor registers is that a binary number of all 0’s when decremented, produces a result of all 1’s.

Arithmetic Instructions:

- Addition (ADD) , Subtract (SUB), Multiply (MUL), Divide (DIV) These instructions may be available for different types of data .it may be binary, decimal, floating-point data. The mnemonics for three add instructions that specify different data types are shown below: ADDI:-Add two binary integer numbers. ADDF:-Add two floating-point numbers. ADDD:-Add two decimal numbers in BCD.

Arithmetic Instructions:

- Add with Carry (ADDC):- A special carry flip- flop is used to store the carry from an operation. The instruction “add with carry” performs the addition on two operands plus the value of the carry from the previous computational.
- Subtract with borrow(SUBB):- subtracts two words and a borrow which may have resulted from a previous subtract operation.
- Negate (2’s complement):-the negate instruction forms the 2’s complement of a number effectively.

Logical Instructions

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

Logical Instructions

They are useful for manipulating individual bits or a group of bits that represent binary-coded information. The logical instruction consider each bit of the operand separately and treat it as a boolean variable.

Logical Instructions

Clear(CLR):- the clear instruction causes the specified operand to be replaced by 0's.

Complement (COM):-the complement instruction produces the 1's complement by inverting all the bits of the operands.

The AND,OR,XOR instructions produces the corresponding logical operations on individual bits of the operands.

Logical Instructions

Clear Carry (CLRC), Set Carry (SETC), Complement Carry (COMC): Individual bits such as a carry can be cleared, set,or complement with appropriate instructions.

Enable interrupts (EI), Disable interrupts (DI) Flips flops that controls the interrupts facility and is either enabled or disabled by means of bit manipulation instructions.

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 through carry	RORC
Rotate left through carry	ROLC

Shift instructions

Instructions to shift the content of an operand Shifts are operations in which the bits of a word are moved to the left or right. the bit shifted in a end of the word determines the type of shift used. Shift instruction may specify either Logical shifts, Arithmetic shifts ,or rotate

Shift instructions

Logical shift right (SHR) Logical shift LEFT(SHL): Logical shift inserts 0 to the end bit position. the end position is the leftmost bit for shift right and the rightmost bit position for the shift left.

Arithmetic Shift right (SHRA), Arithmetic Shift left (SHLA): Arithmetic shift right instruction must preserve the sign bit in leftmost position.

Shift instructions

Rotate right(ROR), Rotate left (ROL): Rotate instructions produces a circular shift. Bits shifted out at one end of the word are not lost as in a logical shift but are circulated back into the other end.

Shift instructions

Rotate right(ROR), Rotate left (ROL): Rotate instructions produces a circular shift. Bits shifted out at one end of the word are not lost as in a logical shift but are circulated back into the other end.

Program Control Instructions

A program control instruction changes address value in the PC and hence the normal flow of execution change in PC causes a break in the execution of instructions.

It is an important feature of the computers since it provides the control over the flow of the program and provides the capability to branch to different program segments.

Name	Mnemonic
Branch	BR
Jump	JMP
Skip	SKP
Call	CALL
Return	RET

Program Control Instructions

- Branch (BR) and Jump (JMP) instructions are used sometimes interchangeably but, they are different.
- Usually Jump is used to refer to unconditional version of branch.
- Skip (SKP) instructions is used to skip one(next) instruction. It can be conditional or unconditional. It does not need an address field.
- In skip instruction we increment the PC in execution stage, effectively incrementing it by 2.

Conditional Branch Instructions

A conditional branch instruction is a branch instruction that may or may not cause a transfer of control

If the condition is true, control is transferred to the effective address (PC←Add). If the condition is false, the program continues with the next instruction (PC←PC+1).

Mnemonic	Branch Condition	Tested Condition
BZ	Branch if zero	Z = 1
BNZ	Branch if not zero	Z = 0
BC	Branch if carry	C = 1
BNC	Branch if no carry	C = 0
BP	Branch if plus	S = 0
BM	Branch if minus	S = 1
BV	Branch if overflow	V = 1
BNV	Branch if no overflow	V = 0