

# Chapter 5

## Basic Computer Organization and Design

# Instruction Codes

- ❖ The internal organization of a digital system is defined by the sequence of micro-operations it performs on data stored in its registers.
- ❖ A program is a set of instructions that specify the operations, operands, and the sequence by which processing has to occur.
- ❖ The computer reads each instruction from memory and places it in a control register. The control then interprets the binary code of the instruction and proceeds to execute it by issuing a sequence of micro-operations.
- ❖ An **Instruction Code** is a group of bits that instructs the computer to perform a specific operation.

The most basic part of an instruction code is its **operation part**.

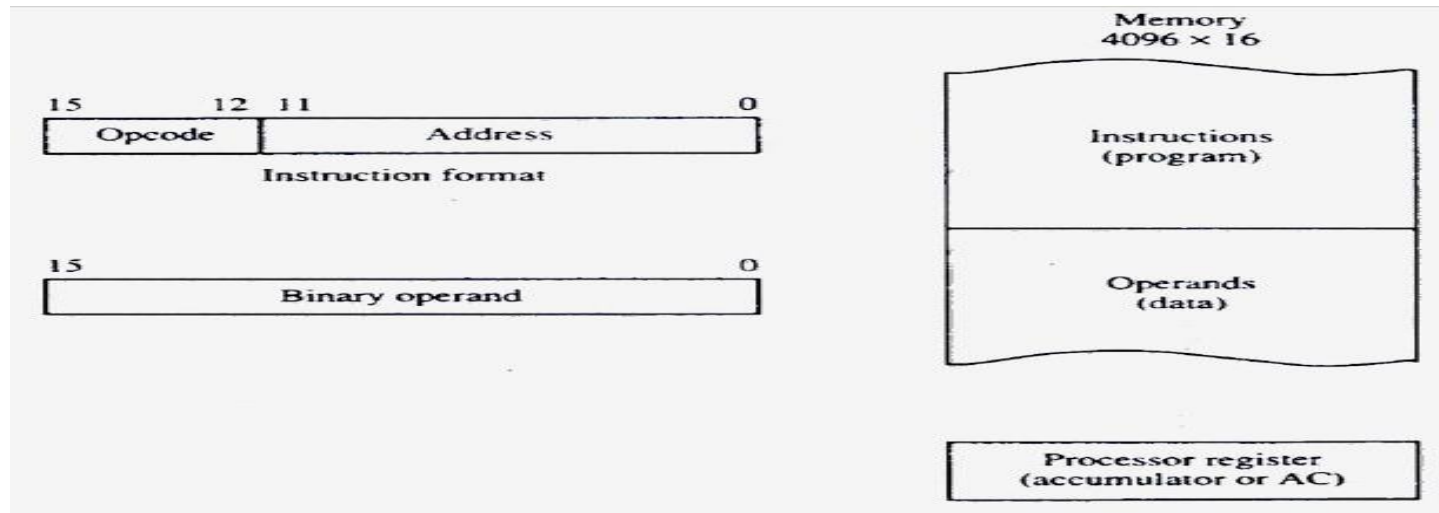
# CONT...

The operation code of an instruction is a group of bits that defines such operations as **Add**, **Subtract**, **Multiply**, **Shift** and **Complement**. The operation part of an instruction code specifies the operation to be performed.

## 5.1.1 Stored Program Organization

- ▶ The simplest way to organize a computer is to have *one processor register* and *an instruction code* format with two parts. The first part of instruction code specifies the *operation* to be performed and the second specifies *an address*. The memory address tells the control where to find an operand in memory.
- ▶ Instructions are stored in one section of memory and data in another.

# CONT...



## Stored program organization

**Note:** Computers that have a single processor register usually assign to the term Accumulator (AC).

# Computer Registers

- ▶ Computer instructions are usually stored in consecutive memory locations and executed sequentially one at a time.
- ▶ The control reads an instruction from specific address in memory and executes it.
- ▶ It is also necessary to provide a register in a control unit for storing the instruction code after it is read from the memory.
- ▶ The data register (DR) holds the operand read from memory.
- ▶ The accumulator (AC) register is a general purpose processing register.
- ▶ The instructions read from memory are placed in instruction register (IR).
- ▶ The temporary register (TR) is used for holding temporary data during the processing.

# CONT...

Register Symbol	Number of Bits	Register Name	Function
DR	16	Data register	Holds memory operand
AR	12	Address Register	Holds address of memory
AC	16	Accumulator	Processor register
IR	16	Instruction Register	Holds instruction code
PC	12	Program Counter	Holds address of instruction
TR	16	Temporary Register	Holds temporary data
INPR	8	Input Register	Holds input character
OUTR	8	Output Register	Holds Output character

List of registers for the basic computer

# CONT...

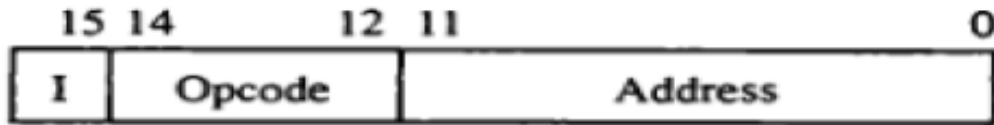
- ▶ The basic computer has eight registers, a memory unit, and a control unit. Paths must be provided to transfer information from one register to another and between memory and registers
- ▶ The memory address register (AR) has 12 bits since this is the width of a memory address.
- ▶ The program counter (PC) also has 12 bits and it holds the address of the next instruction to read from memory after the current instruction is executed.
- ▶ The PC goes through a counting sequence and causes the computer to read sequential instructions previously stored in memory.

# Computer Instructions

- ▶ The basic computer has three instruction code formats.
- ▶ Each format has 16 bits.
- ▶ The operation code (opcode) part of the instruction contains three bits and the meaning of the remaining depends on the operation code encountered.
- ▶ A memory-reference instruction uses 12 bits to specify an address and one bit to specify the addressing mode I ( $I = 0$  for direct address and  $I = 1$  for Indirect address).
- ▶ The register-reference instructions are recognized by the operation code 111 with a 0 in the left most bit (bit 15) of the instruction.
- ▶ it also specifies an operation on or a test of the AC register.

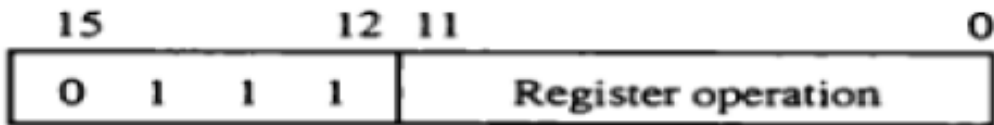


# CONT...



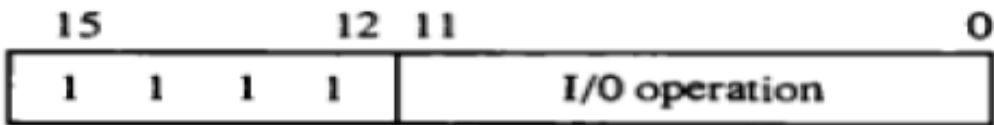
(Opcode = 000 through 110)

(a) Memory – reference instruction



(Opcode = 111, I = 0)

(b) Register – reference instruction



(Opcode = 111, I = 1)

(c) Input – output instruction

Instruction format of basic computer.

# Instruction Set Completeness

- ▶ A computer should have a set of instructions so that the user can construct machine language program to evaluate any function that is known to be computable.
- ▶ The set of instructions are said to be complete if the computer includes a sufficient number of instructions in each of the following categories:
  - ▶ 1. Arithmetic, Logical and Shift Instructions.
  - ▶ 2. Instructions for moving information to and from memory and processor registers.
  - ▶ 3. Program control instructions together with instructions that check status conditions.
  - ▶ 4. Input and Output instructions.

# Timing and Control

- ▶ In order to control the steps of the instruction cycle, it is necessary to introduce a counter, whose output is used as input to the control logic. Sequence Counter Register (SC) is a register that holds a count value, can be reset/cleared to zero and can be incremented (or decremented).



# CONT...

- ▶ The timing and control unit is the component that determines what the ALU should do at a given instant.
- ▶ There are two kinds of control organization:
  1. Hardwired Control
  2. Microprogrammed Control

**Hardwired control:** The control logic is implemented with digital circuits (decoders, flip-flops, etc.).

**Microprogrammed control:** Control information is stored in a control memory. Required modifications can be done by updating the micro-program in control memory.

# Instruction Cycle

- ▶ Fetch an instruction from memory
- ▶ Decode the instruction
- ▶ Read the effective address from memory if the instruction has an indirect address
- ▶ Execute the instruction

# Design of Basic Computer

- ▶ The basic computer consists of the following hardware components;
- ▶ A memory unit with 4096 words of 16-bits each
- ▶ Nine registers: AR, PC, DR, AC, IR, TR, OUTF, INPR, and SC
- ▶ Seven flip-flops: I, S, E, R, IEN, FGI, and FGO (to hold 1-bit of information)
- ▶ Two decoders: a 3x8 operation decoder and a 4x16 timing decoder
- ▶ A 16-bit common bus
- ▶ Control logic gates

# design of accumulator logic

- ▶ The adder and logic circuit has three sets of input.
- ▶ One set of 16 input comes from output of AC.
- ▶ an other set of 16input comes from the data register DR.
- ▶ a third set of input eight input comes from the input register INPR the output of the adder and logic circuit provides the data input for the register,
- ▶ in addition, it is necessary to include logic gate controlling the LD,INR and CLR in the register and for controlling the operation of the adder and logic circuit.
- ▶ in order to design the logic associated with AC, it is necessary to go over the register transfer statements

# Central Processing Unit.

- ▶ The part of the computer that performs the bulk of data processing operations is called the Central processing unit and is referred to as the CPU.
- ▶ The CPU is made up of three major parts, The register set stores intermediate data used during the execution of the instructions.
- ▶ The Arithmetic and Logic unit (ALU) performs the required micro-operations for executing the instructions.
- ▶ The control unit supervises the transfer of information among the registers and instructs the arithmetic and logic units as to which operation to perform.



# CONT...

Computer architecture includes:

- the instruction formats,
- addressing modes,
- the instruction sets and
- the general organizations of the CPU registers



Figure 5-1: Major components of CPU

# General Register Organizations .

- ▶ because memory access is the most time consuming operation in a computer. It is more convenient and more efficient to store these intermediate values in processor register.
- ▶ When a large number of registers are included in the CPU, It is most efficient to connect them through a common bus system.
- ▶ The registers communicate with each other not only for direct data transfers, but also while performing various micro operations.
- ▶ The operation selected in the ALU determines the arithmetic or Logic micro-operation that is to be performed. The result of micro-operation is available for output data and also goes in to the inputs of all the registers

# CONT...

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.

For example, to perform the operation  $R1 \leftarrow R2 + R3$ , 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 selector (SELB): to place the content of R3 into bus B.
3. ALU operation selector (OPR): to provide the arithmetic add  $A + B$ .
4. Decoder destination selector (SELD): to transfer the content of output bus in to R1.

# Control Word

There are 14 binary selection inputs in the unit, and their combined value specifies a control word.

Binary code	SEL.A	SEL.B	SEL.D
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

# CONT...

The ALU provides arithmetic and logic operation. In addition, the CPU must provide shift operations.

A control word of 14-bits is needed to specify a micro-operation in the CPU. The control word for a given micro-operation can be derived from the selection variables.

<b>OPR Select</b>	<b>Operation</b>	<b>Symbol</b>
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

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

# cont...

The increment and transfer micro-operations do not use the B input of the ALU. For these cases, the B field is marked with a dash.

Micro-operation	Symbolic Designation				Control Word
	SELA	SELB	SELD	OPR	
$R1 \leftarrow R2 - R3$	R2	R3	R1	SUB	010 011 001 00101
$R4 \leftarrow R4 \vee R5$	R4	R5	R4	OR	100 101 100 01010
$R6 \leftarrow R6 + 1$	R6	-	R6	INCA	110 000 110 00001
$R7 \leftarrow R1$	R1	-	R7	TFSA	001 000 111 00000
Output $\leftarrow$ R2	R2	-	None	TFSA	010 000 000 00000
Output $\leftarrow$ Input	Input	-	None	TFSA	000 000 000 00000
$R4 \leftarrow \text{sh1 } R4$	R4	-	R4	SHLA	100 000 100 11000
$R5 \leftarrow 0$	R5	R5	R5	XOR	101 101 101 01100

examples of micro operation for the cpu

# Stack Organization

- ▶ A useful feature that is included in the CPU of most computers is a stack or last-in, first-out (LIFO) list.
- ▶ A stack is a storage device that stores information in such a manner that the item stored last is the first item retrieved.
- ▶ The register that holds the address for the stack is called a stack pointer (SP) because its value always points at the top in the stack.
- ▶ The two operations of a stack are the insertion and deletion of items.
- ▶ The operation of the insertion is called PUSH (or push-down)
- ▶ The operation of deletion is called POP (or pop-up)

# Register Stack.

- ▶ A stack can be placed in a portion of a large memory or it can be organized as a collection of a finite number of memory words or registers.

block diagram of 64 word stack





# Cont...

DR is the data register that holds the binary data to be written in to or read out of the stack.

EMPTY is set to 1 when the stack is empty of items.

The PUSH operation is implemented with the following sequence of micro-operations  
SP ← SP + 1 Increment stack pointer  
M[SP] ← DR Write item on top of the stack  
If (SP = 0) then (FULL = 1) Check if stack is full  
EMPTY = 0  
Mark the stack not empty

The stack pointer is incremented so that it points to the address the next higher word.

# Cont...

A memory write operation inserts the word from DR in to the top of the stack.

- ▶ The pop operation consists of the following sequences of microoperations:
- ▶  $DR \leftarrow M[SP]$  Read item from the top of the stack
- ▶  $SP \leftarrow SP - 1$  Decrement stack pointer If  $(SP = 0)$  then  $(EMPTY = 1)$  Check if stack is empty  $(FULL = 0)$  Mark the stack not full
- ▶ The top item is read from the stack into DR.

# Cont...

- The implementation of a stack in the CPU is done by assigning a portion of memory to stack operation and using a processor register as a stack pointer.
- a portion of computer memory partitioned into three segments:
  - ✓ Program,
  - ✓ Data, and
  - ✓ Stack.
- The program counter PC points at the address of the next instruction in the program.
- The address register ER points at an array of data. The stack pointer SP points at the top of the stack.
- PC is used during the fetch phase to read an instruction.
- AR is used during the execute phase to read an operand.
- SP is used to push or POP items into or from the stack.

# Cont...

Most computers do not provide hardware to check for stack overflow (full stack) or underflow (empty stack).

The two micro-operations needed for either the push or pop are:

1. An access to memory through SP and
2. Updating SP.

# Instruction Formats

- ▶ A computer will usually have a variety of instruction code formats.
- ▶ The bits of the instruction are divided into groups called Fields.
- ▶ 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 designate a memory address or a processor register
  3. A mode field that specifies the way the operand or the effective address is determined
- ▶ The number of address field in the instruction format of a computer depends on the internal organization of its registers.
- ▶ Most computers fall into one of three types CPU organizations.
  1. Single accumulator organization
  2. General register organization
  3. Stack organization

# Cont...

- ▶ In an accumulator-type organization, all operations are performed with an implied accumulator register.
- ▶ the instruction that specifies an arithmetic addition is defined by an assembly language:
  - ▶ ADD X, where X is the address of the operand.
  - ▶ Which results  $AC \leftarrow AC + M[X]$
  - ▶ AC is the accumulator register and M[X] symbolizes the memory word located at address X.

# Cont...

- ▶ The instruction format in a computer with a general register organization type needs two or three register address fields.  
e.g `ADD R1, R2, R3`  $R1 \leftarrow R2 + R3$ .
- ▶ The number of address field in the instruction can be reduced from three to two if the destination register is the same as one of the source register.  
e.g `ADD R1, R2` denotes  $R1 \leftarrow R1 + R2$ .
- Only register addresses for R1 and R2 need be specified in this instruction.  
e.g `ADD R1, X` denotes  $R1 \leftarrow R1 + M[x]$ .
- It has two address fields, one for register R1 and the other for the memory address X.
- ▶ Computers with multiple processor registers use the move instruction with a mnemonics `MOV` to symbolize a transfer instruction.  
e.g `MOV R1, R2` denotes  $R1 \leftarrow R2$  (or  $R2 \leftarrow R1$ ) depending on the particular computer.

# Cont...

- ▶ Thus, transfer-type instructions need two address fields to specify the source and the destination.
- ▶ Computers with stack-organization would have PUSH and POP instructions which require an address field.
  - e.g PUSH X will push the word at address X to the top of the stack.
- ▶ Operation-type (e.g ADD) instructions do not need an address field in stack-organized computers.
- ▶ This is because the operation is performed on the two items that are on top of the stack.



# Cont...

We will use symbols ADD, SUB, MUL and DIV for four arithmetic operations; and LOAD and STORE for transfers to and from memory and AC register.

## Three-address Instruction

- ▶ Computers with three-address instruction formats can use each address field to specify either a processor register or a memory operand.
- ▶ The program in assembly language that evaluates  $X = (A+B) * (C+D)$  is shown below, together with comments that explain the register transfer operation of each instruction.
- ▶ ADD R1, A, B      $R1 \leftarrow M[A] + M[B]$
- ▶ ADD R2, C, D      $R2 \leftarrow M[C] + M[D]$
- ▶ MUL X, R1,R2      $M[X] \leftarrow R1 * R2$

# Cont...

- ▶ The advantage of three-address format is that it results in short programs when evaluating arithmetic expressions.
- ▶ The disadvantage is that the binary-coded instructions require too many bits to specify three addresses.

# Two- Address Instruction

- ▶ 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 shown 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
```

# One-Address Instruction

- ▶ One-address instructions use an implied accumulator (AC) register for all data manipulations.

- ▶ 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.

# Zero-Address Instruction

- ▶ 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.

# RISC Instructions

- ▶ RISC stands for reduced instruction set computer.
- ▶ A program for a RISC-type CPU consists of LOAD and STORE instructions that have one memory and one register address and computational-type instructions that have three addresses with all three specifying processor registers.
- ▶ program to evaluate  $X = (A+B)(C+D)$ .

## Addressing Mode

- ▶ The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually referenced. 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.

# Cont...

- ▶ 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

# Direct and Indirect Addressing Modes

- ▶ It is sometimes convenient to use the address bits of an instruction code not as an address but as an actual operand.
- ▶ When the second part of an instruction code specifies an operand, the instruction is said to have an **immediate operand**. When the second part specifies the address of an operand, the instruction is said to have **direct address**.
- ▶ This is in contrast to a third possibility called **indirect address**, where the bits in the second part of the instruction designate an address of memory word in which the address of the operand is found.



# Cont...

**Implied Mode:** In this mode the operands are specified implicitly in the definition of the instruction.

Examples for Implied Mode are”

- the instruction “complement accumulator”
- all register reference instructions that use an accumulator
- Zero-address instructions in stack organized

**Immediate Mode:** In this mode the operand is specified in the instruction itself.

an immediate mode instruction has an operand field rather than an address field.

The operand field contains the actual operand to be used in conjunction with the operation specified in the instruction.

Immediate mode instructions are useful for initializing registers to a constant value.

# Cont...

- ▶ When the address field specifies a processor register, the instruction is said to be in the register mode.

**Register Mode:** In this mode the operands are in registers that reside within the CPU.

**Register Indirect Mode:** In this mode the instruction specifies a register in the CPU whose contents give the address of the operand in memory.

the selected register contains the address of the operand rather than the operand itself. \

# Cont...

- The advantage of the register indirect mode instruction is that the address field of the instruction uses fewer bits to select a register than would have been required to specify a memory address directly.

## **Autoincrement and Autodecrement Mode:**

- This is similar to the register indirect mode except that the register is incremented or decremented after (or before) its value is used to access memory.
- The address field of an instruction is used by the control unit in CPU to obtain the operand from memory.

## **Direct Addressing Mode:**

- In this mode the effective address is equal to the address part of the instruction.

# Cont...

## **Indirect Addressing Mode:**

- In this mode the address field of the instruction gives the address where the effective address is stored in memory.
- Control fetches the instruction from memory and uses its address part to access memory again to read the effective address.

## **Relative Addressing mode:**

- In this mode the content of program counter is added to the address part of the instruction in order to obtain the effective address.
- The address part of the instruction is usually a signed number (in 2's complement representation) which can be either negative or positive.

# Data Transfer and Manipulation

- ▶ Computers provide an extensive set of instruction to give the user the flexibility to carry out various computational tasks.
- ▶ The instruction set of different computers differ from each other mostly in the way the operands are determined from the address and mode fields
- ▶ Most computer instructions can be classified in to three categories.
  1. Data transfer instruction.
  2. Data manipulation instruction
  3. Program control instruction

## **Data transfer instruction.**

- Data transfer instructions move data from place to place in the computer to another without changing the data content. Example Load,Store,Move,Exchange,Input,Output,Pop and Push.

# Data Manipulation

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

# Cont...

## Arithmetic Instruction

- The four basic arithmetic operations are addition, subtraction, multiplication and division.
- Most computers provide instructions for all four operations which are:
  - Increment
  - ✓ Decrease
  - ✓ Add
  - ✓ Subtract
  - ✓ Multiply and
  - ✓ Divide,
- Add with carry, Subtract with Borrow and Negate(2's complement).

# Logical and Bit manipulation Instructions

- Logical instructions perform binary operations on strings of bits stored in registers.
- They are useful for manipulating individual bits or a group of bits that represent binary-coded information.
- Some logical and bit manipulation instructions are:
  - ✓ Clear,
  - ✓ Complement,
  - ✓ AND,
  - ✓ OR,
  - ✓ EX-OR,
  - ✓ Set Carry,
  - ✓ Clear Carry,
  - ✓ Complement Carry, Enable Interrupt and Disable Interrupt



# Shift Instruction

- It is an Instruction 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 at the end of the word determines the type of shift used.
- Shift instructions may specify either logical shifts, arithmetic shifts, or rotate type operations.
- In either case the shift may be to the right or to the left.

# Cont...

Example:

- ✓ Logical Shift Right
- ✓ Logical Shift Left
- ✓ Arithmetic Shift-Left
- ✓ Rotate Right
- ✓ Rotate Left
- ✓ Rotate Right Through Carry and
- ✓ Rotate Left Through Carry

# Reduced Instruction Set computer (RISC)

- An important aspect of computer architecture is the design of the instruction set for the processor.
- The instruction set chosen for a particular computer determines the way that machine language programs are constructed.
- (RISC) are, that computers use fewer instructions with simple constructs so they can be executed much faster within the CPU without having to use memory as often.

# RISC characteristics

- ▶ Relatively few instructions and Relatively few addressing modes
- ▶ Memory access limited to load and store instructions
- ▶ All operations done within the register of the CPU
- ▶ Fixed length, easily decoded instruction format
- ▶ Single cycle instruction execution
- ▶ Hard-wired rather than micro-programmed control
- ▶ A relatively large number of registers in the processor
- ▶ Use overlapped register windows to speed up procedure call and return
- ▶ Efficient instruction pipeline
- ▶ Compiler support for efficient transmission high-level language programs into machine language programs

# CISC characteristics

- ▶ A computer with a large number of instructions is classified as a complex instruction set computer, abbreviated as CISC.
- ▶ Large number of instructions
- ▶ Some instructions that perform specialized tasks and are used infrequently
- ▶ A large variety of addressing modes – typically from 5 to 20 different modes
- ▶ Variable length instruction formats
- ▶ Instructions that manipulate operands in memory RISC characteristics