Computer System Architecture
UNIT – 2
INSTRUCTIONS & INSTRUCTION SEQUENCING

Syllabus

- ~~Fundamentals of Instructions~~
- Operands
- Op Codes
- Instruction Formats
- Addressing Modes

Operands

Operand Types

- Machine operations depend on the types of data being processed.
- The data, as far as the machines are concerned, are in the form of 0s and 1s.
- But to categorize and use with assembly and high-level language programs, the following pattern can be used.
  - Addresses
    - Numbers that represent specific locations in the memory
  - Numbers
    - Integer(or Fixed-point), Floating-point, Decimal
  - Characters (alphanumeric)
    - Sequence of bits represents each character: ASCII, IRA, EBCDIC
  - Logical data (true or false situation)
    - Bit oriented view of data

Op Codes

Op Code Types

- Data transfer
  - MOVE, STORE, LOAD, EXCHANGE, CLEAR, SET, PUSH, POP
- Arithmetic
  - ADD, SUB, MUL, DIV, ABS, NEG, INCR, DECR
- Logical
  - AND, OR, NOT, XOR, SHIFT, ROTATE, TEST, COMPARE
- Control transfer
  - JUMP, RET, EXEC, SKIP, HALT, WAIT, NOP
- Input/ Output control
  - INPUT, OUTPUT, START I/O, TEST I/O
- Conversion
  - TRANSLATE, CONVERT
- System control
  - RESERVED FOR OS

Instruction Formats

Components

- Operation Code (Op code)
  - Operation to be performed
  - Example: ADD
- Source Operand
  - Inputs for Op Code
  - 0 or more source operands can be there in an instruction
- Resultant Operand
  - Result is stored at the location specified by this operand
- Next Instruction Reference
  - Location from where to fetch the next instruction
  - Example: PC+1, PC-1, JUMP, LOOP
- Refer: https://www.learncomputerscienceonline.com/instruction-format/

Instruction Formats

Instruction Set Based on Number of Addresses

- Computers may have instructions of varying length containing varying number of addresses.
- The number of address fields in the instruction depends on the internal organization of registers.
- This gives rise to three, two, one, or zero address instructions.
- To explain the effect of the number of addresses on the computer program, we will use the following arithmetic statement for all of the above four instructions.

$$X = (A+B)*(C+D)$$

Instruction Formats

Three address instructions

- Computers using three address instructions, have an address field each to specify either processor register or a memory address from where operands can be fetched.
- Taking the above example, we give below the equivalent mnemonic codes.

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

- where RI and R2 are the two registers; MIA] means that the operand is the memory address represented by A and so on.
- While the three address instruction has the advantage of short programs in using arithmetic expression, specifying three addresses using too many bits makes it a disadvantage.

Instruction Formats

<u>Two address instructions</u>

- Most commonly used instructions on commercial computers.
- Here also, the address field can specify a register or a memory address.
- Now, the program to evaluate the expression can be rewritten as:

```
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
```

Instruction Formats

One address instructions

- These use a SPR called Accumulator(AC) for all data manipulation
- Now, the program to evaluate the expression can be rewritten as:

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

Instruction Formats

Zero address instructions

- Used in Stack Based Organization
- Stack does not require address field for ADD and MUL
- PUSH and POP need address field that specify the operand
- Now, the program to evaluate the expression can be rewritten as:

PUSH A	TOS $\leftarrow$ A
PUSH B	TOS $\leftarrow$ B
ADD	TOS $\leftarrow$ (A+B)
PUSH C	TOS $\leftarrow$ C
PUSH D	TOS $\leftarrow$ D
ADD	TOS $\leftarrow$ (C+D)
MUL	TOS $\leftarrow$ (C+D) * (A+B)
POP X	M[X] $\leftarrow$ TOS

Addressing Modes

- To specify a memory address in an instruction word, the most obvious technique is simply to give the address in binary form.
- This is called direct addressing.
- Although direct addressing provides the most straightforward (and fastest) way to give a memory address, several other techniques are also used.
- The use of one of these techniques is generally motivated by one of the following considerations:
    1. **Desire to shorten address section**
    2. **Programmer convenience**
    3. **System operation facilities**
- In many computer systems, the computer will have several different programs in memory at a given time and will alternate the running of these programs.
- To efficiently load and remove these programs from the memory in different locations, addressing techniques are provided which make the program relocatable, meaning that the same program can run in many different sections of the memory.

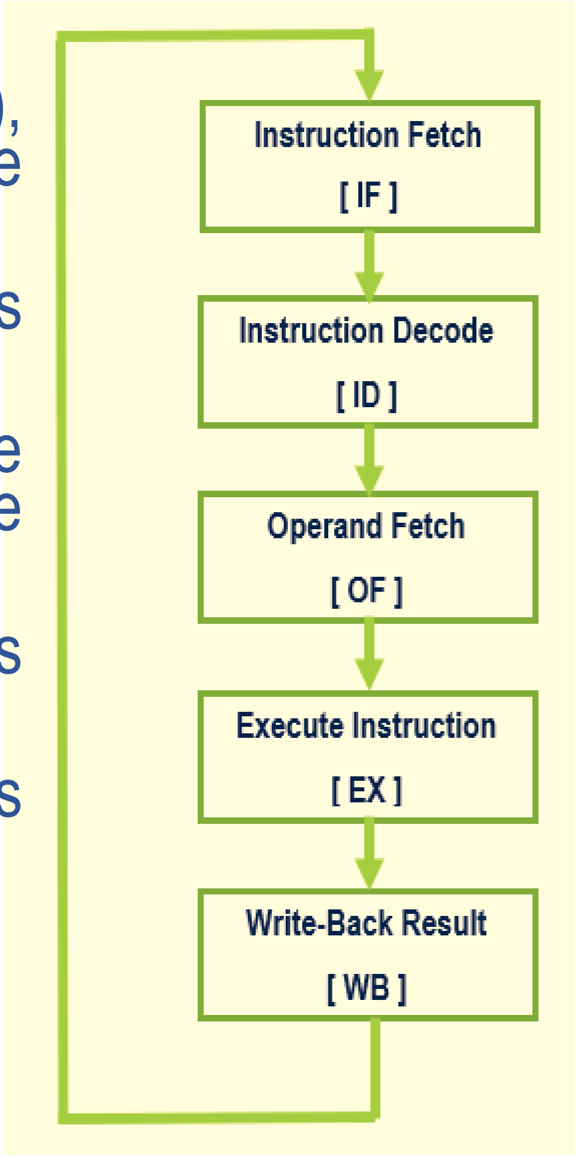
Addressing Modes

- As mentioned earlier, the operand field of an instruction specifies the address from where the data has to be fetched.
- This may be a memory address or one of the computer registers.
- The way the operands are chosen is dependent on the addressing mode of the instruction.
- The addressing mode specifies a rule for interpreting or translating the address field of the instruction into an effective address from where the operand is actually referenced.
- This is done to accommodate the following provisions:
  1. To give the programmer the versatility of providing facilities like pointers to memory, counter for loop control, data indexing, and program relocation.
  2. To reduce the number of bits in the addressing field of the instruction depending on the addressing mode.

Addressing Modes

Instruction Cycle

- The computer has a register called the Program Counter (PC), which keeps track of program instructions that are stored in the memory.
- It holds the address of the next sequential instruction and is incremented every time an instruction is fetched.
- Decoding done in step 2 will determine the operation to be performed, the addressing mode to be used and the address of the operand;
- Step 3 will use the addresses provided by step 2, to fetch operands either from the memory or the registers.
- In step 4, the actual operation will be performed using the operands so fetched.
- In Step 5, the result will be put back into the operand address.
- This process is repeated for the next instruction of the program,



Addressing Modes

- The instructions are defined with a variety of addressing modes, which may be single or a combination of two or more addressing modes.
- The mode field of the instruction is used to locate operands needed to perform the operation specified by the opcode,
- There may or may not be an address field in the instruction.
- If there is an address field, it may refer to a memory address or a processor register.
- Various Addressing Modes:
  - Immediate Addressing
  - Direct Addressing
  - Indirect Addressing
  - Displacement Addressing
    - Relative Addressing
    - Base Register Addressing
    - Indexing
  - Stack Addressing

Addressing Modes

- Immediate Addressing
  - Value of the operand is (immediately) available in the instruction itself   ADD #5
- Direct Addressing
  - Address of the memory location that holds the operand   ADD 1000
- Indirect Addressing
  - Address of the memory location that holds the address of the operand   ADD (1000)
- Register Addressing
  - Name of a register that holds the operand   ADD R1
- Register Indirect Addressing
  - Name of a register that holds the address of the operand   ADD (R1)
- Displacement Addressing
  - Index Addressing
    - Address of the operand is obtained by adding a constant to the content of a register   ADD $X(R_{index})$
  - Relative Addressing
    - Same as indexed addressing except that the program counter (PC) replaces the index register   ADD X(PC)
  - Base Register Addressing
- Stack Addressing
  - Instruction need not include any address. Operation is done on the TOS   ADD