Computer Architecture - Lecture 4

- **Intel 86 processor (an example CISC ISA):**
  - Intel 8086/8088
    - Address lines = 20 lines (memory address space = 2^{20} byte = 1Mbyte
    - Size of MAR = 20 bits
    - Data bus lines (MDR size) = 8 lines in 8080
    - 16 lines on 8086, we can access two bytes at a time in 8086. But in 8088 two bytes are accessed in two cycles.
- **Microprocessor vs Micro controller:**
  - Microprocessor (CPU on a chip) (General purpose computing)
  - Micro controller (CPU + memory) (I/O interface on a chip) (Embedded systems)
- **Registers in 8086/8088:**
  - General purpose group: AX(accumulator), BX(base), CX(counter), DX(data)
    - Each is 16 bits and can be divided into two 8 bits register.
  - Index and pointer registers:
    - SI (source index)
    - DI (destination index)
    - SP (stack pointer)
    - BP (base pointer)
    - IP (instruction pointer) is the offset of program counter.
    - Each is **16** bits (not divided)
  - Segment registers (for memory segmentation):
    - To hold the address of memory segments
    - CS (code segment) machine code (instructions)
    - DS (Data segment) لو عددها كبير (global/static data)
    - ES (Extra data segment)
    - SS (stack segment) (local variables)
    - Each is **16** bits (Not divided)
- **Physical memory address:** (20 bits address)
  - Represented as: <segment address> : <offset address>

    16 bits 16 bits
  - Physical address = (segment address << 4) + offset address.
  - << 4 → shift 4
- The full address (20 bits address) is not used inside the program but only offset address.
- The o/s is responsible of setting the segment address.

- A memory location (variable) with physical address = 30, what could be its segment and offset components?
- **ANS:**
  - Segment address x 16 + offset address = 30
  - 0 x 16 + 30 = 30
  - 1 x 16 + 14 = 30
  - This address can either be represented as 0 : 30 or 1 : 14.
- The physical address 300 can be represented as 0 : 300, 1 : 284, 2 : 268.
- Segment size = 2^{16} = 64k byte, but segments overlap.
- **Basic instruction in 8086/8088 processor:**
  - Data movement instructions:
    - mov instructions:
      - General syntax : mov <destination>, <source>
        - Copies source in destination
        - mov AX,BX → AX = BX
      - Constraints:
        - Source destination must be of the same size. ~~mov AX,BL~~
  - Destination can be register or memory operands.
    Source can be a register or memory or immediate.
    Destination cannot be immediate.
    - mov AX, 130 (immediate operand)
    - but mov 30, AX (False)
    - mov [30], AX (store AX in memory location with address 30) (load memory word in address 30 into AX)
  - Destination and source cannot be simultaneously memory operands (we cannot have more than one memory operand in mov instructions)
    - mov [30], [70]
    - **Forms of mov instructions:**
      - mov m, r
      - mov m, i
      - mov r, r
      - mov r, m
      - mov r, i
- In intel word = 16 bits = 2 bytes.
- In MIPS word = 32 bits = 4 bytes.
- **Arithmetic and logic operations:**
  - Add, susb, mul, div → performs integer arithmetic operations.
  - And, or, not, xor → logic operations.
  - Shl, shr, rol, ror, sar → shift and rotate.
  - Rcl→ rotate left through carry flag.

    - Rcr → rotate right through carry.
- Add AX, BX → AX = AX + BX
- Ex:
    - mov AX, 70

      → Adds 70+30, stores result in AX.
    - Add AX, 30
- Ex:
    - mov AL, 01101011b
    - And AL, 10110110b
- **mul instruction:**
    - mul <source>
        - if the source size is 8 bits then destination is AL, result will be in AX.
    - Ex: mov BL, 7

      mov AL, 9

      mov BL
        - AX (63) = AL x BL
- Write a program to compute Z (AX) = X (DL) * Y (DH) + 30
    - mov AL, DL (X)

      mul DH (Y)   (AX = X * Y)

      Add AX, 30
- If the source size is 16 bits, the mul instruction will multiply source AX and store the result DX (high word) : AX (low word)
- Ex:
    - mov AX, 30

      mov BX, 40

      mul BX
        - AX = 1200
        - DX = 0
- **Div instruction:**
    - Div <source>
        - If the source size is 8 bits: divide source by AL stores result in AL and remainder in AH.
        - mov AX, 23
        - mov BL, 5
        - Div BL
            - Divides AX by BL (23/5), result in AL = 4 (remainder in AH = 3)
    - If the sourse is 16 bits:
        - Divides DX:AX by the source stores result in AX and remainder in DX

- Ex:
    - mov DX, 0
      mov AX, 23
      mov BX, 5
      div BX
        - AX = 4, DX =3