Lecture 4

1 8086 Instruction Set

1.1 8086 Instruction Classes

Category	Brief Overview	Examples
Data Allocation	Pseudo instructions are used in Assembly to allocate memory	DB, DW, DD, DQ
Data Transfer	*instructions used to move single bytes and words between a register, a memory location, or an I/O port.*	Mov, In, Out, Push, Pop, XCHG
Arithmetic	Perform Arithmatic operation	Add, Sub, Mul, Div
Bit Manipulation	Perform logicals, shifts, and rotates on bits	AND, OR, XOR, NOT, SHL, SHR, ROL, ROR
Processor Control	controlprocessor operation and change flags	CLC, LOCK, WAIT
Unconditional Transfer	change program flow	jmp, call
Conditional Branch	change program flow conditionally	jz, jnz, jc
Interrupt	Executes a specific interrupt service routine	INT
String	string manipulation	movsw, MOvSw

8086 instructions do not permit memory-to-memory operations [Except for string processing]

1.2 Data Allocation

- DB pesudocode is used to define a single byte
  - MYVAR DB 45H → single byte containing 45H
  - MYARR DB 20H,-250,40H,'A'
- DW is used for 16-bit word, DD is used for 32-bit word, DQ is used for 64-bit word
- DUP allows to allocate multiple bytes.
  - DB ?, ?, ?, ?, ? ↔ DB 5 DUP(?)

- EQU directive does not allocate any memory: it creates a constant value to be used by Assembler
  - CR EQU 13; Carriage return
- Labels in front of the directives remember offsets from the beginning of the segment

```
value DW 100                    ; value 0-1 initialized to 100
sum DD 50                       ; sum 2-5 initialized to 50
marks DW 10 DUP (?)             ; marks 6-25 Uninitialized
message DB 'The grade is:',0    ; message 26-39
char1 DB ?                      ; char1 40
```

1.3 MOV Instruction [Data Transfer Instructions]

MOV DEST,SRC	SRC and DEST can be any combination of registers and memory **EXCEPT MEM,MEM**
MOV CX , DX ;	DX → CX [**Register address mode**]
MOV AH, BL;	AH ← BL
MOV AX, 2025H ;	2025 → AX [**Immediate mode**]
MOV BX, [memLabel];	move the contents of the 20-bit address computed from DS:START to BX [**Memory Direct Addressing**]
MOV CH, [BX] ;	DS:BX → CH [**Register indirect Addressing**] effective address of a memory operand may be taken directly from one of the base or index registers (BX, BP, SI, DI) the effective address is estimated ***by default*** from the contents of register **DS:BX, SS:BP, DS:SI, or ES:DI**
MOV *Dsip*[BP], CX ;	(CL) → (SS:(BP)+*Dsip*) and (CH) → (SS:(BP)+*Dsip*+1)
	Low address goes to the low byte. the effective address is calculated from the sum of a displacement value and the contents of register SI or DI [**Indexed Addressing**]
MOV AX, 4 [BX] [SI];	(DS:(BX)+(SI)+4) → AX the effective address is computed from the sum of a base register (BX or BP), an index register (SI or DI), and a displacement This mode is used to access two-dimensional arrays

1.4 Arithmetic Instructions

1.4.1 8086 Multiplication

```
mul     src     ;acc := acc * src,
                ;unsigned 8-bit or 16-bit operations
imul    src     ;acc := acc * src
                ; Signed multiplication
```

- for byte [AX] = [AL]*[mem/reg]
- for word [DX][AX] = [AX] * [mem/reg]

- For 8-bit*8-bit multiplication, BYTE PTR can be used to identify a byte operation
  - Example: IMUL BYTE PTR[BX]
- For 16-bit * 16-bit, assembler directive WORD PTR can be used
  - Example: MUL WORD PTR[SI]
- MUL is used for
- If (BX)=0050h, (DS)=3000h, (30050h)=0002h, and (AX)=0006h, then, after
  MUL WORD PTR [BX]
  (DX) = 0000h and (AX) = 000Ch

1.4.2 8086 Division

- DIV MEM or DIV REG → unsigned division
  - when operand is a byte: AL = AX / operand AH = remainder (modulus)
  - when operand is a word: AX = (DX AX) / operand DX = remainder (modulus)

```
MOV AX, 203     ; AX = 00CBh
MOV BL, 4
DIV BL          ; AL = 50 (32h), AH = 3
RET
```

- IDIV MEM /IDIV REG → signed division

References

M. RAFIQUZZAMAN, "Fundamentals of Digital Logic and Microcomputer Design," Fifth Edition.