INDIAN INSTITUTE OF TECHNOLOGY ROORKEE

System Software
CSN-252
Introduction

```
        LDA     ALPHA
        ADD     INCR
        SUB     ONE
        STA     BETA
        LDA     GAMMA
        ADD     INCR
        SUB     ONE
        STA     DELTA
          :
ONE     WORD    1
ALPHA   RESW    1
BETA    RESW    1
GAMMA   RESW    1
DELTA   RESW    1
INCR    RESW    1
```

```
        LDS     INCR
        LDA     ALPHA
        ADDR    S, A
        SUB     #1
        STA     BETA
        LDA     GAMMA
        ADDR    S, A
        SUB     #1
        STA     DELTA
        :
ALPHA   RESW    1
BETA    RESW    1
GAMMA   RESW    1
DELTA   RESW    1
INCR    RESW    1
```

- avoids the need to fetch INCR from memory each time it is used in the calculation
- use of immediate addressing for the constant 1
- Memory space / memory references

```
        LDA     ALPHA
        ADD     INCR
        SUB     ONE
        STA     BETA
        LDA     GAMMA
        ADD     INCR
        SUB     ONE
        STA     DELTA
          :
ONE     WORD    1
ALPHA   RESW    1
BETA    RESW    1
GAMMA   RESW    1
DELTA   RESW    1
INCR    RESW    1
```

```
        LDS     INCR
        LDA     ALPHA
        ADDR    S, A
        SUB     #1
        STA     BETA
        LDA     GAMMA
        ADDR    S, A
        SUB     #1
        STA     DELTA
          :
ALPHA   RESW    1
BETA    RESW    1
GAMMA   RESW    1
DELTA   RESW    1
INCR    RESW    1
```

▪ Instruction Fetch	8	9
▪ Operand Fetch	6	3
▪ Instruction Execute	8	9
▪ Store result	2	2

Program to copy one 11-byte character string to another

```
        LDX     ZERO
MOVCH   LDCH    STR1, X
        STCH    STR2, X
        TIX     ELEVEN
        JLT     MOVCH
          :
STR1    BYTE    C'TEST STRING'
:
STR2    RESB    11
ZERO    WORD    0
ELEVEN  WORD    11
```

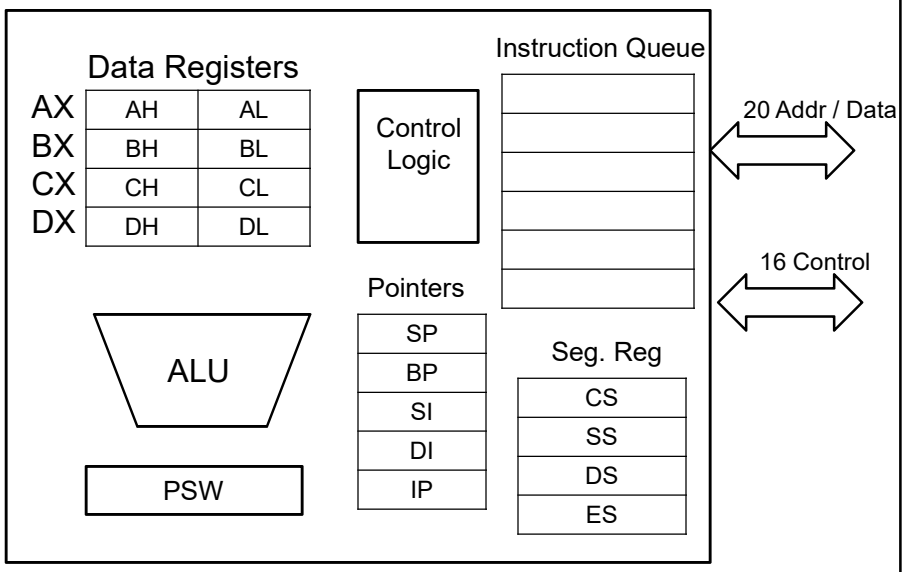
```
        LDT     #11
        LDX     #0
MOVCH   LDCH    STR1, X
        STCH    STR2, X
        TIXR    T
        JLT     MOVCH
          :
STR1    BYTE    C'TEST STRING'
STR2    RESB    11
```

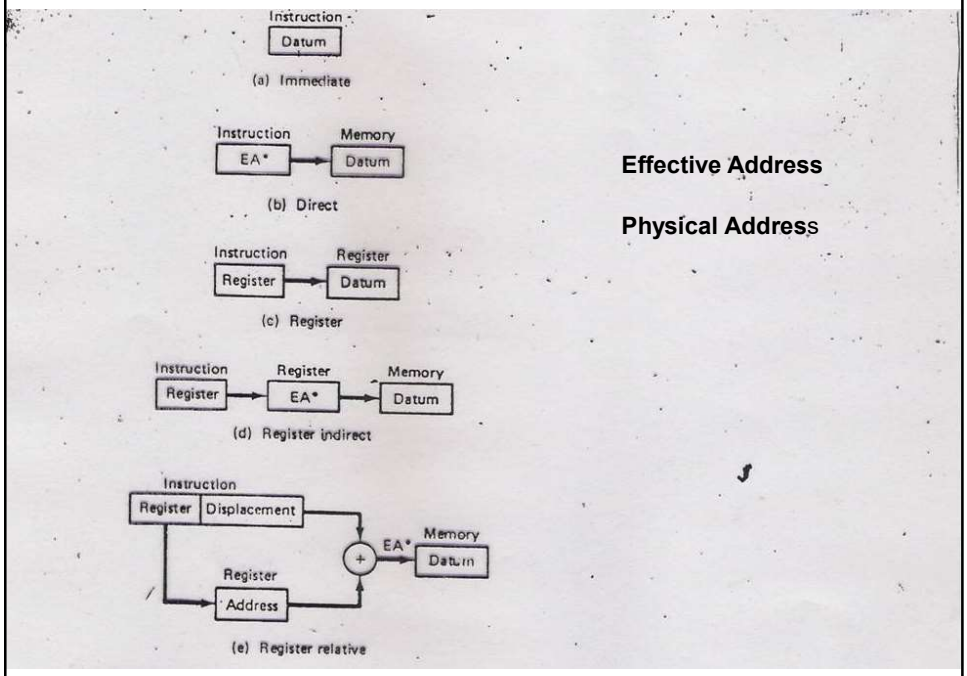
Input and Output

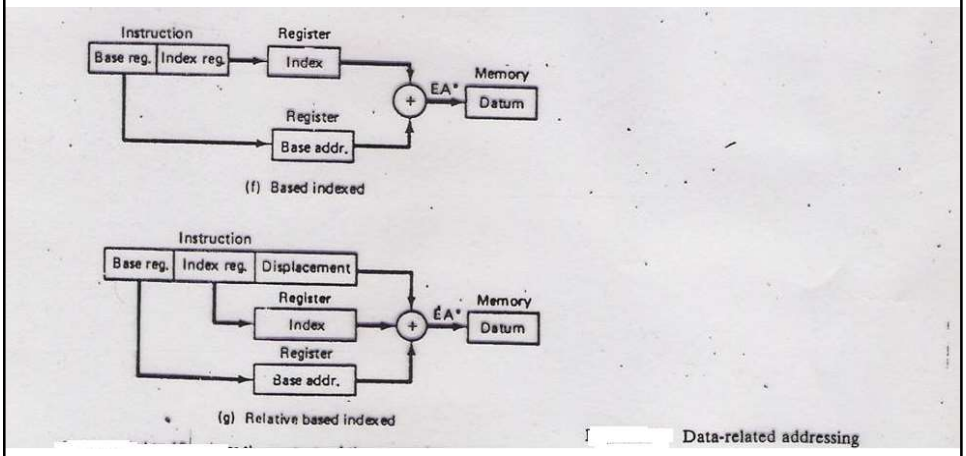
```
INLOOP    TD      INDEV
          JEQ     INLOOP
          RD      INDEV
          STCH    DATA
          .
          .
OUTP      TD      OUTDEV
          JEQ     OUTP
          LDCH    DATA
          WD      OUTDEV
          .
          .
INDEV     BYTE    X'F1'
OUDEV     BYTE    X'05'
DATA      RESB    1
```

Some Examples

8086 Internal Configuration







Data-related addressing

Example

(BX) = 0158 (DI) = 10A5

disp = 1B57 (DS) = 2100

- Direct
- Register indirect assuming BX
- Register relative assuming BX
- Based indexed
- Relative based indexed

8086 Instruction Formats vary from 1 to 6 bytes in length

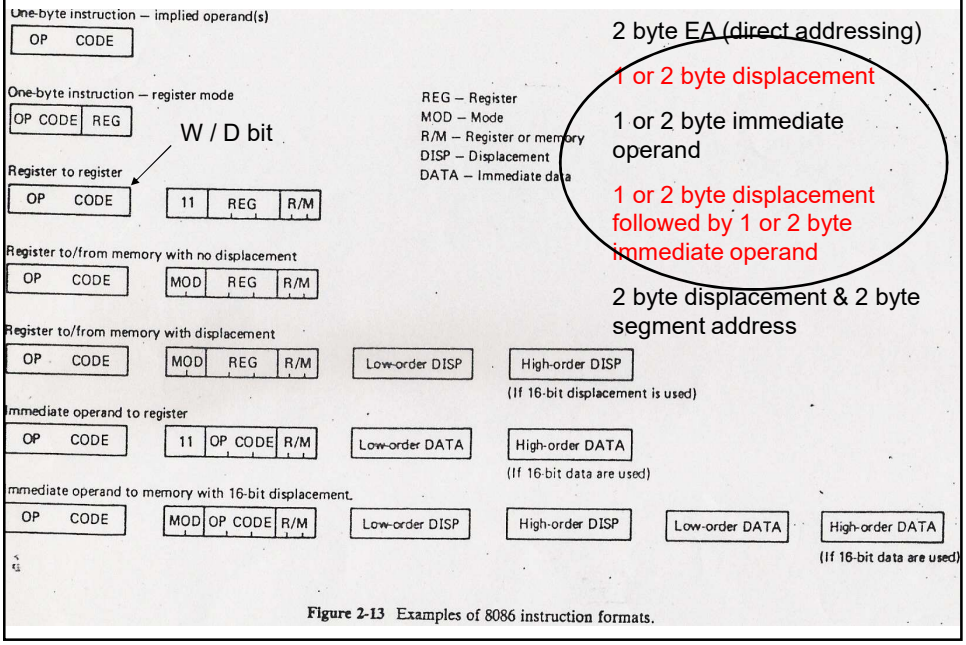


Figure 2-13 Examples of 8086 instruction formats.

- Byte 1: contains opcode, d-bit and w-bit
  - D-bit: Tells the register operand in REG field in byte 2 is source or destination operand
  - W-bit: specifies whether the operation will be performed on 8 bit or 16-bit data
- Byte 2: REG field is used to identify the register for the first operand
  - Example: REG = 0, W = 0 / 1 => AL / AX
  - 2-bit MOD field and 3-bit R/M field specify the second operand

Mode		Displacement
0	0	Disp = 0 Low order and High order displacement are absent
0	1	Only Low order displacement is present with sign extended to 16-bits.
1	0	Both Low-order and High-order displacements are present.
1	1	r/m field is treated as a 'Reg' field.

11	
W = 0	W = 1
AL	AX
CL	CX
DL	DX
BL	BX
AH	SP
CH	BP
DH	SI
BH	DI

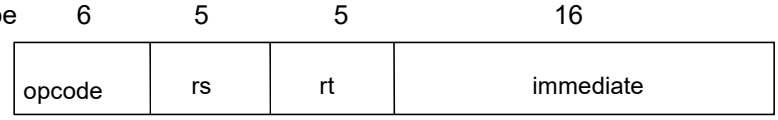
Mode		Displacement
0	0	Disp = 0 Low order and High order displacement are absent
0	1	Only Low order displacement is present with sign extended to 16-bits.
1	0	Both Low-order and High-order displacements are present.
1	1	r/m field is treated as a 'Reg' field.

R/M	Operand Address
0 0 0	EA = (BX) + (SI) + Displacement
0 0 1	EA = (BX) + (DI) + Displacement
0 1 0	EA = (BP) + (SI) + Displacement
0 1 1	EA = (BP) + (DI) + Displacement
1 0 0	EA = (SI) + Displacement
1 0 1	EA = (DI) + Displacement
1 1 0	EA = (BP) + Displacement
1 1 1	EA = (BX) + Displacement

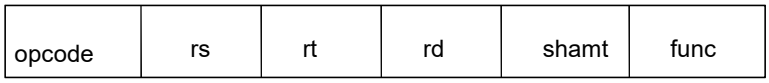
MIPS instruction formats

- addressing modes encoded into the opcode
- fixed length encoding for performance

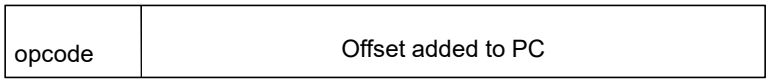
I-type



R-type



J-type



Data Addressing Modes

immediate, displacement → register indirect, absolute addressing

DADDIU R4, R2, #3 Regs[R4] ← Regs[R2]+3

LD R4, 100(R1) Regs[R4] ← Mem[100+Regs[R1]]