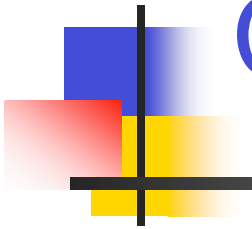
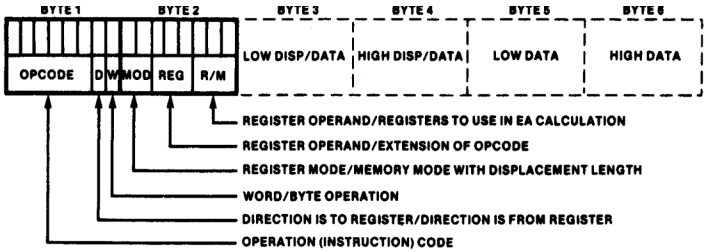


Chapter 4

Machine Language Coding and the Debug Program

Instruction Encoding

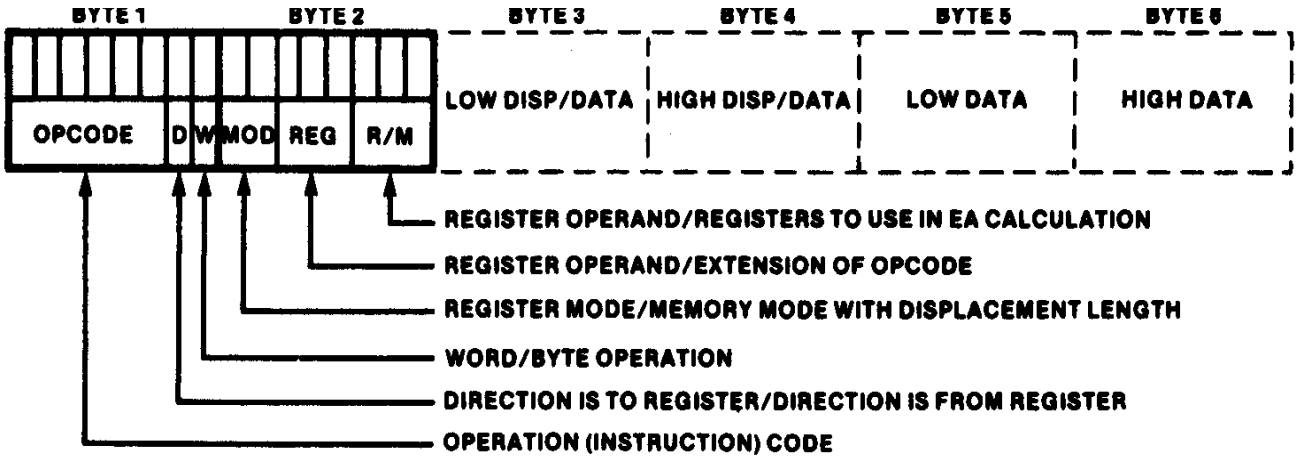


80x86's instruction set is hybrid length

- Multiple instruction sizes, but all have byte wide lengths—
  - 1 to 6 bytes in length for 8088/8086
  - Up to 17 bytes for 80386,80486, and Pentium
- Advantages of hybrid length
  - Allows for many addressing modes
  - Allows full size (32-bit) immediate data and addresses
- Disadvantage of variable length
  - Requires more complicated decoding hardware—speed of decoding is critical in modern uP

Instruction Encoding

- Information encoded in an instruction
  - What operation ?
  - What operands ?
  - Byte, word or double-word ?
  - Operands in register or memory ?
  - How the address is to be generated, if mem?



First Byte of an Instruction

MOV = Move:	7 6 5 4 3 2 1 0
Register/memory to/from register	1 0 0 0 1 0 d w
Immediate to register/memory	1 1 0 0 0 1 1 w
Immediate to register	1 0 1 1 w reg
Memory to accumulator	1 0 1 0 0 0 0 w
Accumulator to memory	1 0 1 0 0 0 1 w
Register/memory to segment register	1 0 0 0 1 1 1 0
Segment register to register/memory	1 0 0 0 1 1 0 0

REG	W = 0	W = 1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

- **Opcode field (6-bits)—specifies the operation to be performed by the instruction**
  - **Move immediate to registers/memory = 1100011**
  - **Move memory to accumulator = 1010000**
  - **Move segment register to register/memory = 10001100**
- **REG (3-bit)—selects a first operand as a register**
  - **Move immediate to register = 1011(w)(reg)—only requires one register which is the destination**
    - **Accumulator register= 000**
    - **Count register = 001**
    - **Data Register = 010**
- **W (1-bit)—data size word/byte for all registers**
  - **Byte = 0, Word =1**
- **D (1-bit)—register direction: tells whether the register which is selected by the REG field in the second byte is the source or destination**
  - **Add register to register = 000000(d)(w)**
  - **D=0 → source operand, D=1 → destination operand**

Example of One-Byte Instruction

INC = Increment:

Register/memory	1 1 1 1 1 1 1 w	mod 0 0 0 r/m	(DISP-LO)	(DISP-HI)
Register	0 1 0 0 0 reg			
AAA = ASCII adjust for add	0 0 1 1 0 1 1 1			
DAA = Decimal adjust for add	0 0 1 0 0 1 1 1			

REG	W = 0	W = 1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

- **One Byte Example:**
  - **Encode the instruction in machine code**

    `INC CX`
  - **Solution:**
    - **Use "INC register" instruction format—special short form for 16-bit register**

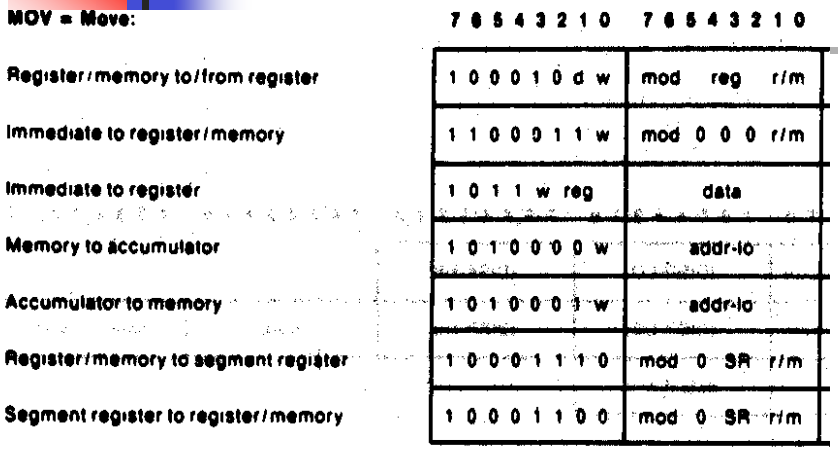
      01000 (REG)
    - **CX is destination register**

      CX = 001
    - **Machine code is**

      01000 (001) = 01000001 = 41H → one byte instruction

      `INC CX` = 41H

Second Byte of an Instruction



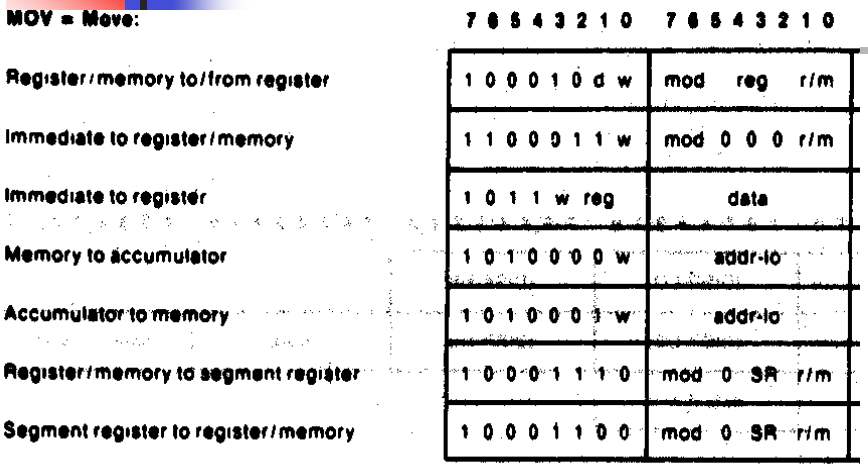
MOV = Move:	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Register/memory to/from register	1 0 0 0 1 0 d w	mod reg r/m
Immediate to register/memory	1 1 0 0 0 1 1 w	mod 0 0 0 r/m
Immediate to register	1 0 1 1 w reg	data
Memory to accumulator	1 0 1 0 0 0 0 w	addr-lo
Accumulator to memory	1 0 1 0 0 0 1 w	addr-lo
Register/memory to segment register	1 0 0 0 1 1 1 0	mod 0 SR r/m
Segment register to register/memory	1 0 0 0 1 1 0 0	mod 0 SR r/m

CODE	EXPLANATION
00	Memory Mode, no displacement follows*
01	Memory Mode, 8-bit displacement follows
10	Memory Mode, 16-bit displacement follows
11	Register Mode (no displacement)

*Except when R/M = 110, then 16-bit displacement follows

- **Byte 2 information:**
  - **MOD (2-bit mode field)—specifies the type of the second operand**
    - **Memory mode: 00, 01,10—Register to memory move operation**
      - **00 = no immediate displacement (register used for addressing)**
      - **01 = 8-bit displacement (imm8) follows (8-bit offset address)**
      - **10 = 16-bit displacement (imm16) follows (16-bit offset address)**
    - **Register mode: 11—register to register move operation**
      - **11 = register specified as the second operand**

Second Byte of an Instruction (Cont.)



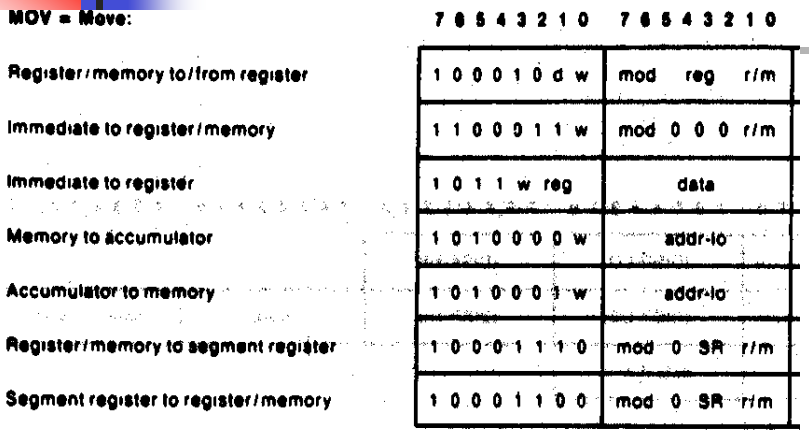
MOV = Move:

	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Register/memory to/from register	1 0 0 0 1 0 d w	mod reg r/m
Immediate to register/memory	1 1 0 0 0 1 1 w	mod 0 0 0 r/m
Immediate to register	1 0 1 1 w reg	data
Memory to accumulator	1 0 1 0 0 0 0 w	addr-lo
Accumulator to memory	1 0 1 0 0 0 1 w	addr-lo
Register/memory to segment register	1 0 0 0 1 1 1 0	mod 0 SR r/m
Segment register to register/memory	1 0 0 0 1 1 0 0	mod 0 SR r/m

REG	W = 0	W = 1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

- **Byte 2 information (continued):**
  - **REG (3-bit register field)—selects the register for a first operand, which may be the source or destination**
    - **Accumulator register= 000**
    - **Count register = 001**
    - **Data Register = 010**
    - **Move register/memory to/from register**
      - **Byte 1= 100010(d)(w)**
      - **Byte 2 = (mod) (reg) (r/m)**
- **Affected by byte 1 information:**
  - **W (1-bit)—data size word/byte for all registers**
    - **Byte = 0**
    - **Word =1**

Second Byte of an Instruction (Cont.)



MOV = Move:

	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Register/memory to/from register	1 0 0 0 1 0 d w	mod reg r/m
Immediate to register/memory	1 1 0 0 0 1 1 w	mod 0 0 0 r/m
Immediate to register	1 0 1 1 w reg	data
Memory to accumulator	1 0 1 0 0 0 0 w	addr-lo
Accumulator to memory	1 0 1 0 0 0 1 w	addr-lo
Register/memory to segment register	1 0 0 0 1 1 1 0	mod 0 SR r/m
Segment register to register/memory	1 0 0 0 1 1 0 0	mod 0 SR r/m

MOD = 11		
R/M	W = 0	W = 1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

- **R/M (3-bit register/memory field)—specifies the second operand as a register or a storage location in memory**
  - **Dependent on MOD field**
    - **Mod = 11 R/M selects a register**
      - **R/M = 000 Accumulator register**
      - **R/M= 001 = Count register**
      - **R/M = 010 = Data Register**
  - **Move register/memory to/from register**
    - **Byte 1= 100010(d)(w)**
    - **Byte 2 = (mod) (reg) (r/m)**
- **Affected by byte 1 information:**
  - **W (1-bit)—data size word/byte for all registers**
    - **Byte = 0**
    - **Word =1**
  - **D (1-bit)—register direction for first operand in byte 2 (reg)**
    - **D = 0 → source operand**
    - **D= 1 → destination operand**

Second Byte of an Instruction (Cont.)

MOV = Move:	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Register/memory to/from register	1 0 0 0 1 0 d w	mod reg r/m
Immediate to register/memory	1 1 0 0 0 1 1 w	mod 0 0 0 r/m
Immediate to register	1 0 1 1 w reg	data
Memory to accumulator	1 0 1 0 0 0 0 w	addr-lo
Accumulator to memory	1 0 1 0 0 0 1 w	addr-lo
Register/memory to segment register	1 0 0 0 1 1 1 0	mod 0 SR r/m
Segment register to register/memory	1 0 0 0 1 1 0 0	mod 0 SR r/m

EFFECTIVE ADDRESS CALCULATION			
R/M	MOD = 00	MOD = 01	MOD = 10
000	(BX) + (SI)	(BX) + (SI) + D8	(BX) + (SI) + D16
001	(BX) + (DI)	(BX) + (DI) + D8	(BX) + (DI) + D16
010	(BP) + (SI)	(BP) + (SI) + D8	(BP) + (SI) + D16
011	(BP) + (DI)	(BP) + (DI) + D8	(BP) + (DI) + D16
100	(SI)	(SI) + D8	(SI) + D16
101	(DI)	(DI) + D8	(DI) + D16
110	DIRECT ADDRESS	(BP) + D8	(BP) + D16
111	(BX)	(BX) + D8	(BX) + D16

- **MOD = 00, 01, or 10 selects an addressing mode for the second operand that is a storage location in memory, which may be the source or destination**
  - **Dependent on MOD field**
    - **Mod = 00 R/M**
      - **R/M = 100 → effective address computed as**

        EA = (SI)
      - **R/M= 000 = → effective address computed as**

        EA = (BX)+(SI)
      - **R/M = 110 = → effective address is coded in the instruction as a direct address**

        EA = direct address = imm8 or imm16

Example: MOV

MOV = Move:	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Register/memory to/from register	1 0 0 0 1 0 d w	mod reg r/m
Immediate to register/memory	1 1 0 0 0 1 1 w	mod 0 0 0 r/m
Immediate to register	1 0 1 1 w reg	data
Memory to accumulator	1 0 1 0 0 0 0 w	addr-lo
Accumulator to memory	1 0 1 0 0 0 1 w	addr-lo
Register/memory to segment register	1 0 0 0 1 1 1 0	mod 0 SR r/m
Segment register to register/memory	1 0 0 0 1 1 0 0	mod 0 SR r/m

EFFECTIVE ADDRESS CALCULATION			
R/M	MOD = 00	MOD = 01	MOD = 10
000	(BX) + (SI)	(BX) + (SI) + D8	(BX) + (SI) + D16
001	(BX) + (DI)	(BX) + (DI) + D8	(BX) + (DI) + D16
010	(BP) + (SI)	(BP) + (SI) + D8	(BP) + (SI) + D16
011	(BP) + (DI)	(BP) + (DI) + D8	(BP) + (DI) + D16
100	(SI)	(SI) + D8	(SI) + D16
101	(DI)	(DI) + D8	(DI) + D16
110	DIRECT ADDRESS	(BP) + D8	(BP) + D16
111	(BX)	(BX) + D8	(BX) + D16

- **Move register/memory to/from register**
  - **Byte 1= 100010(d)(w)**
  - **Byte 2 = (mod) (reg) (r/m)**
- **Affect of byte 1 information:**
  - **W (1-bit)—data size word/byte for all registers**
    - **Byte = 0**
    - **Word =1**
  - **D (1-bit)—register direction for first operand (REG) in byte 2**
    - **D = 0 → source operand**
    - **D= 1 → destination operand**

Example: INC CL

INC = Increment:

Register/memory	1 1 1 1 1 1 1 w	mod 0 0 0 r/m	(DISP-LO)	(DISP-HI)
Register	0 1 0 0 0 reg			
AAA = ASCII adjust for add	0 0 1 1 0 1 1 1			
DAA = Decimal adjust for add	0 0 1 0 0 1 1 1			

MOD = 11		
R/M	W = 0	W = 1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

- **Two byte example using R/M field for a register:**
  - **Encode the instruction in machine code**

    `INC CL`
  - **Solution:**
    - **Use "INC register/memory" instruction format—general form for 8-bit or 16-bit register/memory**
    - **Byte 1**

      1111111(W)
      - **CL= byte wide register → W = 0**

        11111110 =FEH

Example: INC CL

INC = Increment:

Register/memory	1 1 1 1 1 1 1 w	mod 0 0 0 r/m	(DISP-LO)	(DISP-HI)
Register	0 1 0 0 0 reg			
AAA = ASCII adjust for add	0 0 1 1 0 1 1 1			
DAA = Decimal adjust for add	0 0 1 0 0 1 1 1			

MOD = 11		
R/M	W = 0	W = 1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

- **Two Byte example using R/M field for a register (continued):**
  - **Byte 2**

    (MOD) 000 (R/M)
    - **Destination is register CL**
      - **MOD = 11**
      - **R/M = 001**

      (11)000(001) = 11000001 =C1H
  - **Machine code is**

    (Byte 1)(Byte 2) = 11111110 11000001 = FEC1H → two byte instruction

    `INC CL` = FEC1H

Example: MOV BL, AL

DATA TRANSFER

MOV = Move:	76543210	76543210	76543210	76543210	76543210	76543210
Register/memory to/from register	100010dw	mod reg r/m	(DISP-LO)	(DISP-HI)		
Immediate to register/memory	1100011w	mod 000 r/m	(DISP-LO)	(DISP-HI)	data	data if w = 1
Immediate to register	1011w reg	data	data if w = 1			
Memory to accumulator	1010000w	addr-lo	addr-hi			
Accumulator to memory	1010001w	addr-lo	addr-hi			
Register/memory to segment register	10001110	mod 0 SR r/m	(DISP-LO)	(DISP-HI)		
Segment register to register/memory	10001100	mod 0 SR r/m	(DISP-LO)	(DISP-HI)		

MOD = 11		
R/M	W = 0	W = 1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

- **Two byte example using R/M field for a register:**
  - **Encode the instruction in machine code**

    `MOV BL,AL`
  - **Solution:**
    - **Use "register/memory to/from register" instruction format—most general form of move instruction**
    - **Byte 1**

      100010(D)(W)
      - **Assuming AL (source operand) is the register encoded in the REG field of byte 2 (1st register)**
        - **D = 0 = source**
      - **Both registers are byte wide**
        - **W = 0 = byte wide**
      - **Byte 1 = 100010(0)(0) = 10001000 =88H**

Example: MOV BL, AL

DATA TRANSFER

MOV = Move:	76543210	76543210	76543210	76543210	76543210	76543210
Register/memory to/from register	100010dw	mod reg r/m	(DISP-LO)	(DISP-HI)		
Immediate to register/memory	1100011w	mod 000 r/m	(DISP-LO)	(DISP-HI)	data	data if w = 1
Immediate to register	1011w reg	data	data if w = 1			
Memory to accumulator	1010000w	addr-lo	addr-hi			
Accumulator to memory	1010001w	addr-lo	addr-hi			
Register/memory to segment register	10001110	mod 0 SR r/m	(DISP-LO)	(DISP-HI)		
Segment register to register/memory	10001100	mod 0 SR r/m	(DISP-LO)	(DISP-HI)		

REG	W = 0	W = 1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

MOD = 11		
R/M	W = 0	W = 1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

- **Two Byte Example (continued):**
  - **Byte 2**

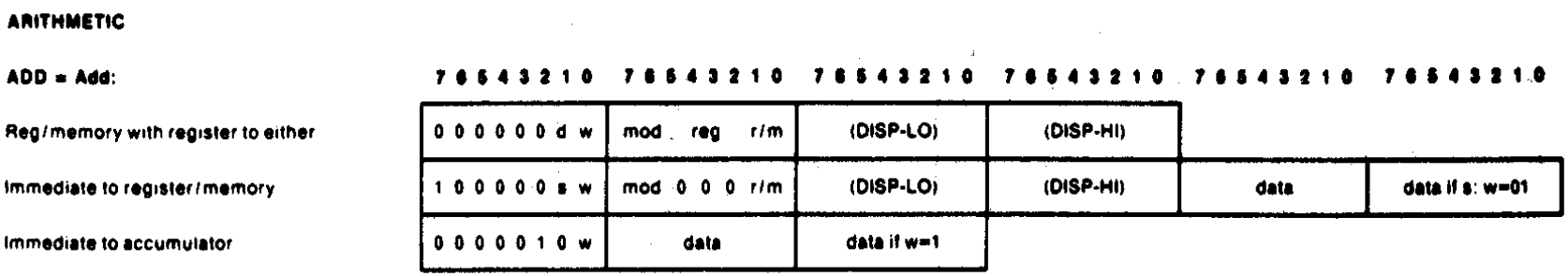
    (MOD)(REG)(R/M)
    - **Both operands are registers**
      - **MOD = 11**
    - **2nd register is destination register BL**
      - **R/M = 011**
    - **1st register is source register AL**
      - **REG = 000**

    (11)000(011) = 11000011 = C3H
  - **Machine code is**

    (Byte 1)(Byte 2) = 10001000 11000011 = 88C3H → two byte instruction

MOV BL, AL = 88C3H

Example: ADD AX, [SI]



ARITHMETIC

ADD = Add:	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Reg/memory with register to either	0 0 0 0 0 0 d w	mod reg r/m	(DISP-LO)	(DISP-HI)		
Immediate to register/memory	1 0 0 0 0 0 s w	mod 0 0 0 r/m	(DISP-LO)	(DISP-HI)	data	data if s: w=01
Immediate to accumulator	0 0 0 0 0 1 0 w	data	data if w=1			

- **Two byte example using R/M field for memory:**
  - **Encode the instruction in machine code**

    `ADD AX,[SI]`
  - **Solution:**
    - **Use "register/memory with register to either" instruction format**
      - **Most general form of add instruction**
      - **No displacement needed—register indirect addressing**
    - **Byte 1**

      000000(D)(W)
      - **AX (destination operand) is the register encoded in the REG field of byte 2 (1st register)**
        - **D = 1 = destination**
      - **Addition is of word wide data**
        - **W = 1 = word wide**
      - **Byte 1 = 000000(1)(1) = 00000011 =03H**

Example: ADD AX, [SI]

EFFECTIVE ADDRESS CALCULATION			
R/M	MOD = 00	MOD = 01	MOD = 10
000	(BX) + (SI)	(BX) + (SI) + D8	(BX) + (SI) + D16
001	(BX) + (DI)	(BX) + (DI) + D8	(BX) + (DI) + D16
010	(BP) + (SI)	(BP) + (SI) + D8	(BP) + (SI) + D16
011	(BP) + (DI)	(BP) + (DI) + D8	(BP) + (DI) + D16
100	(SI)	(SI) + D8	(SI) + D16
101	(DI)	(DI) + D8	(DI) + D16
110	DIRECT ADDRESS	(BP) + D8	(BP) + D16
111	(BX)	(BX) + D8	(BX) + D16

REG	W = 0	W = 1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

- **Two Byte example using R/M field for memory (continued):**
  - **Byte 2**

    (MOD)(REG)(R/M)
    - **Second operand is in memory and pointed to by address is SI**
      - **MOD = 00 → [SI]**
    - **R/M specifies the addressing mode**
      - **R/M = 100 → [SI]**
    - **1st register is destination register AX**
      - **REG = 000**

    (00)000(100) = 00000100 = 04H
  - **Machine code is**

    (Byte 1)(Byte 2) = 00000011 00000100

    = 0304H → two byte instruction

    `ADD AX,[SI] = 0304H`