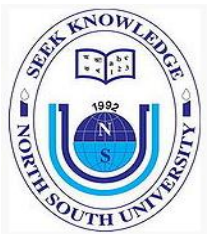


North South University

CSE 331L

Microprocessor Interfacing & Embedded System (Lab)

HOMEWORK 04

SUBMITTED BY:

Name: SADMAN ALAM
ID: 1610544042
Section: 07
Phone Number: 01828157801
E-Mail: sadman.alam@northsouth.edu

SUBMITTED TO:

Lab Instructor's Name: ASIF AHMED NELOY

8086 INSTRUCTION SET

DATA TRANSFER INSTRUCTIONS

MOV - MOV Destination, source

The MOV Instruction copies a word or byte of data from a specified destination. The destination can be a register or memory location. The source can be a register a memory location or an intermediate number. The source and destination can not be both memory locations. They must both the of the same type (bytes or words). MOV instruction does not affect any flag.

- MOV CX, 037AH ; Put immediate number 037AH to CX
- MOV BL, [437AH] ; copy byte in DS at offset 437AH to BL
- MOV AX, BX ; copy content of register BX to AX

▷ MOV DL, [BX] copy byte from memory at [BX] to DL

▷ MOV DS, BX copy word from BX to DS register.

LEA - LEA Register, source

This instruction determines the offset of the variable or memory location named as the source and puts this offset in the indicated 16-bit register. LEA does not affect any flag.

▷ LEA BX, PRICES Load BX with offset of PRICE in DS

▷ LEA BP, SS: STACK_TOP Load BP with offset of STACK_TOP in SS

▷ LEA CX, [BX] [DI] Load CX with EA = [BX] + [DI]

ARITHMETIC INSTRUCTIONS

ADD - ADD Destination, source
ADC - ADC Destination, source

These instructions add a number of some source to a number in some destination and put the result in the specified destination. The ADC also adds the status of the carry flag to the result. The source may be an immediate number, a register or a memory location. The destination may be a register or a memory location.

△ ADD AL, 74H — Add immidieate number 74H to content of AL. Result in AL

△ ADC CL, BL — Add content of BL plus carry status of CL

△ ADD DX, BX — Add content of BX to content of DX

△ ADD DX, [SI] — Add word from memory offset [SI] in DS to content of DX

△ ADC AL, PRICES [BX] — Add byte from effective address PRICES [BX] plus carry status to content of AL

△ ADD AL, PRICES [BX] — Add content of memory at effective address PRICES [BX] to AL

SUB - SUB Destination, source
SBB - SBB Destination, source

These instructions subtract the number in the some source from the number of destination and put the result in the destination. The SBB instruction also subtracts the content of carry flag from the destination. If you first move the byte to a word location such as a 16 bit register and fill the upper byte of the word with 0's.

- SUB CX, BX — CX - BX; Result in CX
- SBB CH, AL — subtract content of AL and content of CF from content of CH. Result in CH
- SUB AX, 3427 H — subtract immediate number 3427 H from AX
- SBB BX, [3427 H] — subtract word at displacement 3427 H in DS and content of CF from BX
- SUB PRICES [BX], 04 H — subtract 04 from byte at effective addres PRICES [BX].
- SBB CX, TABLE [BX] — subtract word from effective addres TABLE [BX] and status of CF from CX
- SBB TABLE [BX], CX — subtract CX and status of CF from word in memory at effective addres TABLE [BX]

MUL - MUL source

This instruction multiples an unsigned byte in some source with an unsigned byte in AL register or an unsigned word in some source with an unsigned word in AX register. The source can be a register on a memory location. You cannot use the CBW instruction for this, because the CBW instruction for this, because the CBW instruction fills the upper byte with copies of the most significant bit of the lower byte.

- MUL BH — Multiply AL with BH; result in AX
- MUL CX — Multiply AX with CX

△ MUL BYTE PTR [BX] Multiply AL with byte in DS pointed to by [BX]

△ MUL FACTOR [BX] Multiply AL with byte at effective address FACTOR [BX], if it is declared as type byte with DB.

△ MOV AX, MCAND_16 Load 16-bit multiplicand into AX

MOV CL, MPLIER_8 Load 8-bit multiplier into CL

MOV CH, 00H set upper byte of CX to all 0's

MUL CX AX times CX; 32-bit result in DX and AX

DIV - DIV source

This instruction is used to divide an unsigned word by a byte or to divide an unsigned double word by a word. When a word is divided by a byte, the word must be in the AX register. The divison can be in a register or a memory location. After the division, AL will contain the 8-bit quotient and AH will contain the 8 bit remainder. If you want to divide a byte by a byte you must first put the dividend by a byte in AL and fill AH with all 0's. If you want to divide a word by another word then put the dividend word in AX and fill DX with 0's.

- DIV BL — Divide word in AX by byte in BL; Quotient in AL, remainder in AH
- DIV CX — Divide down word in DX and AX by word in CX; Quotient in AX and remainder in DX
- DIV SCALE [BX] — AX/[byte at effective address SCALE[BX]] if SCALE [BX] is of type byte

INC - INC Destination

The INC instruction adds 1 to a specified register or to a memory location. AF, OF, PF, SF and ZF are updated but CF is not affected

- INC BL — Add 1 to contain of BL register
- INC CX — Add 1 to contain of CX register

- INC BYTE PTR [BX] — Increment byte in data segment at offset combined in BX
- INC WORD PTR [BX] — Increment the word at offset of [BX] and [BX+1] in Data segment
- INC TEMP — Increment byte or word named TEMP in the data segment.
- INC PRICES [BX] — Increment element pointed to by [BX] in array prices.

DEC - DEC Destination

This instruction subtracts 1 from the destination word or byte. The destination can be a register or a memory location.

△ DEC CL — subtraction 1 from content of CL register

△ DEC BP — Subtract 1 from content of BP register

△ DEC BYTE PTR [BX] — Subtact 1 from byte at offset [BX] in DS

△ DEC WORD PTR [BP] — Subtract 1 from at offset [BP] in SS.

△ DEC COUNT — Subtract 1 from byte or word named count in DS.

DAA (DECIMAL ADJUST AFTER BCD ADDITION)

This instruction is used to make sure the result of adding two BCD numbers is just adjusted to be a legal BCD number.

▷ Let AL = 59 BCD, and BL = 35 BCD

```
ADD AL, BL
DAA
```

AL = 8EH; lower nibble > 9, add 06H to AL

AL = 94 BCD, CF = 0

▷ Let AL = 88 BCD and BL = 49 BCD

```
ADD AL, BL
DAA
```

AL = D1H; AF = 1 add 06H to AL

The DAA instruction updates AF, CF, SF, PF and ZF; but OF is undefined.

AAA (ASCII ADJUST FOR ADDITION)

Numerical data coming into a computer from a terminal is usually in ASCII code. In this code, the numbers 0 to 9 are represented by the ASCII codes 30H to 39H.

D Let AL = 0011 0101 (ASCII 5) and
BL = 0011 1001 (ASCII 9)

ADD AL, BL

AAA

AL = 0110 1110 (6EH, Which is incorrect BCD)

AL = 0000 0100 (unpacked BCD 4)

CF = 1 indicates answer is 14 decimal

LOGICAL INSTRUCTIONS

AND - AND Destination, source

This instruction ANDS each bit in a source byte or word with the same numbered bit in a destination byte or word. The result is put in the specified destination. The content of the specified source is not changed.

- AND CX, [SI] — AND word DS at offset [SI] with word in CX register, Result in CX register
- AND BH, CL — AND byte in CL with byte in BH; Result in BH
- AND BX, 00FFH — 00FFH masks upper byte, leaves lower byte unchanged

OR-OR Destination, source

This instruction ORs each bit in a source byte or word with the same numbered bit in a destination byte or word. The result is put in the specified destination. The content of the specified source is not changed.

▷ OR AH, CL — CL ORed with AH, result in AH, CL not changed

△ OR BP, SI — SL ORed with BP, result in SI, BP, SI not changed

▷ OR SI, BP — BL ORed with immediate number 80H; sets MSB of BL to 1

△ OR CX, TABLE [SI] — CX ORed with word from effective address TABLE [SI]; content of memory is not changed.

XOR - XOR Destination, source

~~The~~ This instruction Exclusive-ORs each bit in a source byte or word with the same numbered bit in a destination byte or word. The result is put in

the specified destination.
The content of the specified destination is not changed.

△ XOR CL, BH — Byte in BH exclusive ORed with byte in CL

△ XOR BP, DI — WORD in DI-ORed with word in BP. Result in BP.

△ XOR WORD PTR [BX], 00FFH — Exclusive-OR immediate number 00FFH with word at offset [BX] in the data segment

CMP-CMP Destination, source

The instruction compares a byte/word in the specified source with a byte/word in the specified destination.

	CF	ZF	SF	Result of subtraction is 0
CX = BX	0	1	0	No borrow required, so CF=0
CX > BX	0	0	0	
CX < BX	1	0	1	Subtraction requires borrow, so CF = 1

△ CMP AL, 01H compare immediate number 01H with byte in AL

△ CMP BH, CL compare byte in CL with byte in BH

△ CMP CX, TEMP Compare word in DS displacement TEMP with word at CX

△ CMP PRICES [BX], 49H compare immediate number 49H with byte at offset [BX] in array PRICES

TEST- TEST Destination, source

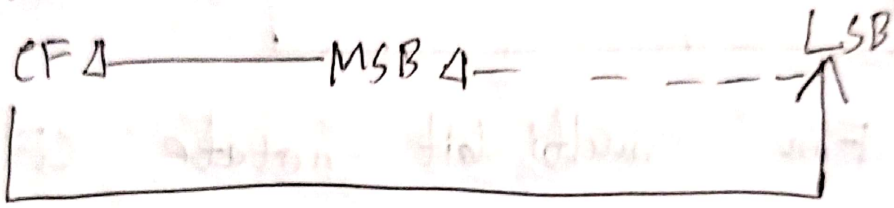
This instruction ANDs the byte/word in the specified source with the byte/word in the specified destination.

- TEST AL, BH — AND BH with AL. No result stored; update PF, SF, ZF.
- TEST CX, 0001H — AND CX with immediate number 0001H. No result stored, update PF, SF, ZF
- TEST BP, [BX] [DI] — AND word one offset [BX] [DI] in DS with word in BP. No result stored. Update PF, SF, ZF

ROTATE AND SHIFF INSTRUCTIONS

RCL - RCL Destination, Count

This instruction rotates all the bits in a specified word on byte some number of bit positions to the left.



For multi-bit rotates, CF will contain the bit most recently rotated out of the MSB.

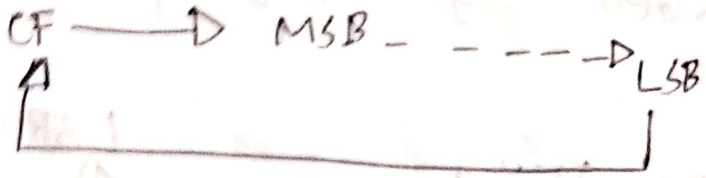
△ RCL DX, 1 — Word in DX 1 bit left, MSB to CF, CF to LSB

△ MOV CL, 4 — Load the number of bit positions to rotate into

RCL SUM [BX], CL — Rotate byte or word of effective addres SUM [BX]

RCR - RCR Destination, count

This instruction rotates all the bits in a specified word or byte some number of bit positions to the right.



For multi-bit rotate, CF will contain the bit most recently rotated out of the LSB.

▷ RCR BX, 1 — word in BX right 1 bit, CF to MSB, LSB to CF

▷ MOV CL, 4
RCR BYTE PTR[BX], 4 — Rotate the byte at offset [BX] in DS 4 positions right

SAL - SAL Destination, count
SHL - SHL Destination, count

SAL and SHL are two mnemonics for same instruction. This instruction shifts each bit in the specified destination some number of bit positions to the left.

CF ← MSB ← - - - - LSB ← 0

SAL BX, 1 — Shift word in BX 1 bit position left, 0 in LSB

MOV CL, 02h — Load desired number of shifts in CL

SAL BP, CL — Shift word in BP left CL bit positions, 0 in LSB

SAL, BYTE PTR [BX], 1 — Shift byte DX at offset [BX] 1 bit position left, 0 in LSB

SAD - SAR Destination, count

This instruction shifts each bit in the specified destination some number of bit position to the right.

▷ SAR DX, 1 Shift word in DI one bit position right, new MSB = old MSB

▷ MOV CL, 02H Load desired number of shifts in CL

SAR WORD PTR [BP], CL shift word at offset [BP] in stack segment right by two bit position.

SHR- SHR Destination, count

This instruction shifts each bit in the specified destination some number of bit positions to the right.

△ SHR BP, 1 shift word in BP one bit position right, 0 in MSB

△ MOV CL, 03H Load desired number of shifts into CL

SHR BYTE PTR [BX] shift byte in DS at offset [BX] 3 bits right, 0's in 3 MSBs

TRANSFER-OF-CONTROL INSTRUCTIONS

JMP (UNCONDITIONAL JUMP TO SPECIFIED DESTINATION)

~~The~~ This instruction will fetch the next instruction from the location specified in the instruction rather than from the next location after the JMP instruction. If the destination is in the same code segment as the JMP instruction then the only instruction pointer will be changed to get the destination location.

JBE/JNA (JUMP IF BELOW OR EQUAL/ JUMP IF NOT ABOVE)

If CF and ZF are both 0 the Instruction will have no effect on program execution.

Δ CMP AX, 4371H
JBE NEXT

compare (AX - 4371H)
JUMP to label NEXT if AX is below or equal to 4371H

Δ CMP AX, 4371H
JNA NEXT

Jump to label NEXT if AX not above 4371H

JG/JUMP IF GREATER/JUMP IF NOT LESS THAN OR EQUAL

This Instruction is usually used after a compare Instruction.

A CMP BL, 39H
JG NEXT

Compare by subtracting 39H from BL.

A CMP BL, 39H
JNLE NEXT

Compare by subtracting ~~39H~~ from BL.

JL / JNGE

A CMP BL, 39H
JL Again

Compare by subtracting ~~1no~~ 39H from BL
Jump to label AGAIN if BL more negative than 39H

A CMP BL, 39H
JNGE AGAIN

Jump to label AGAIN if BL not more positive than equal to 39H

JE/JZ

A: CMP BX, BX — compare (BX - BX)
JE DONE — if yes go to DONE if BX = 0

A IN AL, 30H — Subtract the
SUB AL, 30H — minimum value
JZ START — Jump to label START if the result subtraction is 0

JNE/JNZ

IN AL, 0FBH — read data value the port compare (AL-72)
CMP AL, 72
JNE NEXT

A ADD AX, 0002H — Decrement BX
DEC BX — Jump to label NEXT if BX ≠ 0
JNZ NEXT

STACK RELEASED INSTRUCTIONS

PUSH - PUSH source

- PUSH BX — Decrement SP by 2, copy BX to stack
- PUSH DS — Decrement SP by 2, copy DS to stack
- PUSH BL — Illegal; must push a word
- PUSH TABLE [BX] — Decrement SP by 2 and copy word from memory in DS at EA = TABL + [BX] to stack

POP - POP Destination

△ POP DX copy a word from top of stack to DX; increment SP by 2.

△ POP DS copy a word from top of stack to DS; increment SP by 2.

△ POP TABLE [DX] copy a word from top stack to memory in DS with EA = TABLE + [BX]; increment SP by 2

INPUT - OUTPUT INSTRUCTIONS

IN - IN Accumulator Port

△ IN AL, 0C8 H Input a byte from port 0C8H to AL

△ IN AX, 34H Input a word

from port 34H
to AX.

OUT- OUT Port, Accumulator

- OUT 3BH, AL — copy the content of AL to port 3BH
- OUT 2CH, AX — copy the content of AX to port 2CH

8086 ASSEMBLER DIRECTIVES

ENDS (END SEGMENT)

- CODE SEGMENT
  CODE ENDS

start of logical segment containing code instruction statements

ENDP (END PROCEDURE)

SQUARE — ROOT PROC Start of procedure

SQUARE — ROOT ENDP End of procedure

INCLUDE (INCLUDE SOURCE CODE FROM FILE)

This directive is used to tell the assembler to insert a block of source code from the named file into the current source module.