Memorial University of Newfoundland

# Engineering 4862 MICROPROCESSORS

## Assignment 2

---

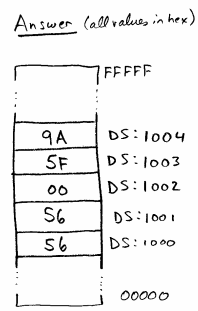
Unless otherwise noted, please show all relevant calculations, and explain your answers where appropriate.

***0. List all 8086/8088 registers that can be accessed as both words and bytes.***

| | | |
|---|---|---|
| Accumulator AX | AH | AL |
| Base BX | BH | BL |
| Counter CX | CH | CL |
| Data DX | DH | DL |

***1. Use a memory map to show the contents of memory locations DS: 1000H to DS: 1004H after all of the following instructions have executed:***

| | **Memory Modification** |
|---|---|
| ***MOV AX, 56H*** | None |
| ***MOV [1001H], AX*** | [DS:1001] ← 56H<br>[DS:1002] ← 00H |
| ***MOV [1003H], 9A5FH*** | [DS:1003] ← 5FH<br>[DS:1004] ← 9AH |
| ***MOV [1000H], AL*** | [DS:1000] ← 56H |




***2. Assume that (CS) = B795H; (DS) = 2000H; (SS) = 0AD4H; (ES) = 30FFH; (SP) = 00FFH; (BP) = 1DF7H; (AX) = 0B24H; (CX) = 1EE4H; (SI) = 3C00H; (DX) = 329FH.***

***a. Calculate the beginning and ending addresses for all of the segments.***

| Segment | Start (Base) | End |
|---|---|---|
| Code, CS=B795H | B7950H | C794FH |
| Data, DS=2000H | 20000H | 2FFFFH |
| Extra, ES=30FFH | 30FF0H | 40FEFH |
| Stack, SS=0AD4H | 0AD40H | 1AD3FH |

***b. Suppose that the offset address for the next instruction to be fetched (that is, the contents of IP) is 902DH. Calculate the physical address from where the next instruction will be fetched.***

Segment: Code, CS=B795H

PA = CS:IP = B795:9020 = C097DH

***c. What will be the contents of BL and AX after the following instruction is executed? Give your results in decimal and hexadecimal.***

***MOV BL, AL***

BL ← AL, AX= 0B24H

So, BL=24H = 36D

AL=24H (unchanged), AX= 0B24H=2852D

***d. Calculate the physical addresses of the memory locations referred to in the following instructions and the contents of all the location(s):***

***AND CX, [1200H]***

[1200] is the memory reference, the PA = DS:1200=21200H. Used to access one word.

***OR ES:[0E8C9H], SI***
ES for Segment override. [0E8C9H] for memory reference. PA = ES:E8C9H = 3F8B9H
Both 3F8B9H and 3F8BAH are referred.

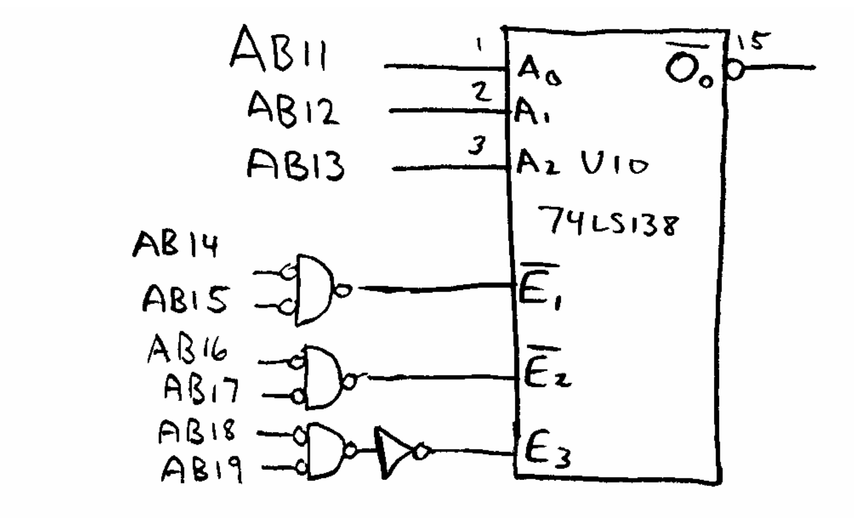
***PUSH DX***
Memory reference is implicit, as the stack is used.
Stack address = SS:SP = 0AD4H:00FFH = 0AE3FH
The memory accessed is not this location, but at 2 less than this. A word is accessed, the location 0AE3DH and 0AE3EH are used.

3. ***What is wrong with, or missing from, each of the following instructions:***
   ***a. MOV ES, 249EH***
   ES is a segment register. It is invalid to assign a segment register directly via an immediate operand.

   ***b. MOV [BX+3EH], 2 (Hint: the use of PTR directive)***
   2 can be either a byte or a word operand. The assembler will complain because it does not know how many bytes the value '2' requires. Use BYTE PTR 2 or WORD PTR 2.

   ***c. MOV [78H], [79H]***
   Attempting to have a memory to memory data transfer. Invalid addressing mode.

4. ***In lab 1, we found that the MUN-88 single board computer has a number of mirror images. Without replacing the 3x8 decoder or the current 2KB SRAM chip, propose a simple scheme to eliminate these mirror images for the RAM and draw a sketch to show that. (Hint: The mirror images are caused by the don't care lines of the address bus. You can use those lines connect to chip select, enable pins of the decoder or SRAM chips).***
   The mirror images are actually caused by the three most significant address lines, which are set to don't cares. One we can give a fixed pattern for those address lines, we will eliminate the confusion. There are many solutions for this question. The diagram below show one possible solution. This requires the addition of one 74LS32 chip (2 input OR gate). The important aspect of this scheme is that AB14 to AB 19 must be all 0s in order to access the RAM. There must be no unused address lines. A similar scheme could be done for the ROM, but all AB16-19 have to be 1.




5. ***Instruction XCHG achieves a swap between the source operand and the destination operand. (Consult the 8086/88 user manual for more detailed information). Assume that the instruction XCHG does not exist in the 8086 instruction set. Write a sequence of instructions to duplicate the instruction XCHG AL,***

***DL. Note that the values in all other registers (including AH and DH) should be their original values when your instruction sequence finishes.***

```
PUSH CX
MOV CL, AL
MOV AL, DL
MOV DL, CL
POP CX
```

Notes: 1. All stack operations are two bytes
2. You can assume a memory location exists to act as the temporary store

6. ***Can the 8-bit input port at location 911 be accessed using direct port I/O addressing? Give an instruction sequence to copy the data from this port to register CL.***
Direct port I/O addressing requires that the address of the I/O port fits into a byte (from 0 to $255_{10}$). Thus $911_{10}$ is too large, and indirect port addressing (Using DX) must be used. Note that Direct/ Indirect I/O addressing is **INDEPENDENT** of the data size of the port (8-bit or 16-bit), and has nothing to do with memory addressing. The instruction should be:

```
MOV DX, 911     ; port address
IN AL, DX       ; Input 8-bit from 911
MOV CL, AL      ; copy data to CL
```

Also note that MOV CL, [911] is wrong! This refers to a memory location, not an I/O port.

7. ***When a CALL is executed, how does the CPU know where to return? What is the difference between a FAR call and a NEAR call?***
The address of the instruction immediately following the CALL is stored on the stack. The last instruction of a called subroutine must be RET in order to the system to pop off the return address from the stack.
In the FAR CALL, both the CS and IP registers are saved on the stack, whereas in a NEAR CALL, only the IP register will be saved on the stack.

8. ***Find the contents of the stack and stack pointer after the execution of the CALL instruction shown next: Assume that SS:1296 right before the execution of CALL and SUM is a NEAR procedure.***

| ***CS:IP*** | |
|---|---|
| ***2450:673A*** | ***CALL SUM*** |
| ***2450:673D*** | ***DEC AH*** |

IP = 673D will be stored in the stack at 1295 and 1294, therefore SS:1295 = 67 and SS:1294 = 3D. And the stack pointer will point to 1294 then.

9. ***Translate one of the following two quotes to 8-bit ASCII format (ignore the names and dates) – (Text book: Section 3.4):***

***a. "640K ought to be enough for anybody." - Bill Gates, 1981***

36 34 30 4B 20 6F 75 67 68 74 20 74
6F 20 62 65 20 65 6E 6F 75 67 68 20
66 6F 72 20 61 6E 79 62 6F 64 79 2E

***b. "I think there is a world market for about 5 computers."***
***– Thomas J. Watson, founder of IBM, 1943***

49 20 74 68 69 6E 6B 20 74 68 65 72 65 20 69 73
20 61 20 77 6F 72 6C 64 20 6D 61 72 6B 65 74 20
66 6F 72 20 61 62 6F 75 74 20 35 20 63 6F 6D 70
75 74 65 72 73 2E

**10. *Write your MUN student number and convert to its unpacked BCD binary equivalent (Text book: Section 3.4).***

Simply match the corresponding number:

| | | |
|---|---|---|
| 0 – 00H (00000000) | 1 – 01H (00000001) | 2 – 02H (00000010) |
| 3 – 03H (00000011) | 4 – 04H (00000100) | 5 – 05H (00000101) |
| 6 – 06H (00000110) | 7 – 07H (00000111) | 8 – 08H (00001000) |
| 9 – 09H (00001001) | | |

eg. Student number : 9972563

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| decimal | 9 | 9 | 7 | 2 | 5 | 6 | 3 |
| BCD | 1001 | 1001 | 0111 | 0010 | 0101 | 0110 | 0011 |
| Or HEX | 09H | 09H | 07H | 02H | 05H | 06H | 03H |

**11. *Find the precise offset location in memory of each ASCII character or data in the following use a memory map (Text book: Section 3.4 for ASCII numbers):***

***ORG 20H***

***Data1 DB 73H, 2FH***

| | | | | | |
|---|---|---|---|---|---|
| DS:0020 | 73 | DS:0021 | 2F | | |

***Data2 DB "737 3527"***

| | | | | | |
|---|---|---|---|---|---|
| DS:0022 | 37 | DS:0023 | 33 | DS:0024 | 37 |
| DS:0025 | 20 | DS:0026 | 33 | DS:0027 | 35 |
| DS:0028 | 32 | DS:0029 | 37 | | |

***ORG 30H***

***Data3 DW 2560H, 101100010101B***

| | | | | | |
|---|---|---|---|---|---|
| DS:0030 | 60 | DS:0031 | 25 | DS:0032 | 15 |
| DS:0033 | 0B | | | | |

***ORG 40H***

***Data4 DD 25684FC4H***

| | | | | | |
|---|---|---|---|---|---|
| DS:0040 | C4 | DS:0041 | 4F | DS:0042 | 68 |
| DS:0043 | 25 | | | | |

***Data5 DQ 7F5EC4527271FEH***

| | | | | | |
|---|---|---|---|---|---|
| DS:0044 | FE | DS:0045 | 71 | DS:0046 | 72 |
| DS:0047 | 52 | DS:0048 | C4 | DS:0049 | 5E |
| DS:004A | 7F | DS:004B | 00 | | |

**12. *It is common practice to save all registers at the beginning of a subroutine. Assume that SP=1288H before a subroutine CALL. Show the contents of the stack pointer and the exact memory contents of the stack after PUSHF, for the following:***

```
1132:0450    CALL PROC1
1132:0453    INC BX
  …….        ……

PROC1        PROC
             PUSH AX
             PUSH BX
             PUSH CX
             PUSH DX
             PUSH SI
             PUSH DI
             …
             …
PROC1        ENDP
```

When the procedure is called, IP, which points to the next instruction to be executed after the CALL, is saved on the stack since it is a NEAR procedure. After the CALL and all PUSH instructions have been executed, the stack is as follows with SP=127A

| | | | |
|---|---|---|---|
| SS:127A | ← | DI | |
| SS:127C | ← | SI | |
| SS:127E | ← | DX | |
| SS:1280 | ← | CX | |
| SS:1282 | ← | BX | |
| SS:1284 | ← | AX | 1285 = AH, 1284 = AL |
| SS:1286 | ← | IP | 1287 = 04, 1286 = 53 |
| SS:1288 | | | |

13. ***The following program adds four words and saves the result. The program contains some errors, fix the errors and make the program run correctly:***

```
TITLE       PROBLEM PROGRAM
PAGE        60, 132
STSEG       SEGMENT
            DB 32 DUP(?)
STSEG       END                     should be ENDS, Segment should end with ENDS
;---------------------------------------------------
DTSEG       SEGMENT
DATA        DW     1234H, 3344H, 5FE2H, 85FAH
            ORG 10H
SUM         DW     ?
DTSG        ENDS                    Label should match
;---------------------------------------------------
CDSEG       SEGMENT
START:      PROC FAR                Without : since it is a non-opcode generating instruction
            ASSUME     CS:CDSEG, DS:DTSEG, SS:STSEG
            MOV DS, DTSEG           Could not move directly, should go through a register
                                    Normally use MOV AX, DTSEG and MOV DS, AX
            MOV CS, 4               Could not move an immediate number to CS register
            MOV BX, 0
            MOV DI, OFFSET DATA
LOOP1       ADD BX, [DI             Loop1 should have a colon :
            INC DI
            DEX BX                  should be DEC
            JNZ    LOOP1
            MOV SI, OFFSET RESULT
            MOV [SI], BX
            MOV AH, 4CH
            INT 21H
CDSEG:      ENDS                    No : for CDSEG
START       ENDP                    This line should be placed before the previous line
            END CDSEG               should be START
```

14. ***Write an Assembly Language Program that summarize eight unsigned byte numbers stored in memory and store the result back to the next memory location. The data segment can be defined as following:***

```
DTSEG               SEGMENT
        Data        DB     23H, 34H, 32H, 45H, 1FH, 27H, 7FH, 90H
        Result      DW     ?
DTSEG               ENDS
```

***(Hint: 1. Use loop to write an efficient code. 2. Pay attention to how to handle the carry bit. You can use a 16-bit register to hold the result, and then you don't need to worry about the carry)***

```
TITLE   ADDING 8 BYTES
PAGE    60, 132
STSEG   SEGMENT
        DB 64 DUP (?)
STSEG   ENDS
;------------------------------------------
DTSEG   SEGMENT
        Data            DB      23H, 34H, 32H, 45H, 1FH, 27H, 7FH, 90H
        Result          DW      ?
DTSEG   ENDS
;------------------------------------------
CDSEG   SEGMENT
MAIN    PROC    FAR
        ASSUME CS:CDSEG, DS:DTSEG, SS:STSEG

        MOV AX, DTSEG
        MOV DS, AX
        MOV CX, 8       ; COUNTER FOR 8 NUMBERS
        MOV SI, OFFSET DATA
        SUB AX, AX      ;CLEAR AX=0
REP:    SUB BX, BX      ;CLEAR BX=0
        MOV BL, [SI]
        ADD AX, BX
        INC SI
        LOOP REP
        MOV RESULT, AX          ;SAVE RESULT BACK

        MOV     AH, 4CH
        INT 21H
MAIN    ENDP
CDSEG   ENDS
        END MAIN
```

Notes: This is just one example, should have many different ways to write this program.