EE 3724 Test #2 - Fall '00 – Reese

Student ID: _____ (no names please)

1. (33 pts) Assume the following memory contents at the start of each of the following instructions Assume the following register contents at the START of each of the following instructions. ALL VALUES in HEX.

ES: 09A0, DS: 09A0, SS: 09A1
AX = A203, DX = 53FE, CX = 00BA, SP = 0018, BP=002E

Give the value of the affected . LIST ALL registers that are affected by the instructions except for the flag registers. Give answers in HEX. For division problems, must identify **quotient** and **remainder** results, and show which registers they are in.

a. (3 pts) sar cl, 2 cl = EEh (sign bit does not change, arithmetic shift right)

b. (3 pts) shr cl,2 cl = 2Eh (shift in a zero via sign bit, logical shift right)

c. (3 pts) shl cl,1 cl = 74h (logical shift left)

d. (2 pts) ror al,4 al = 30h (rotate right 4 positions)

e. (2 pts) xor dl, al dl = FDh

f. (4 pts) mul dl ax = 02FAh ax = al * dl (unsigned) = 3 * 254 = 762 = 02FA h

g. (4 pts) imul dl ax = FFFAh ax = al * dl (signed) = 3 * -2 = -6 = FFFAh

h. (4 pts) div cl al (quotient) = ax / cl (unsigned) = 41475/186 = 222 = DE h
ah (rmdr) = dividend - quotient * divisor = 41475 - (186*222) = 183 = B7h

i. (4 pts) idiv dl al (quotient) = ax/dl (signed) = -24061 / -2 = -12030 (overflow)

j. (4 pts) idiv cl al (quotient) = ax/cl (signed) = -24061/ (-70) = 343 (overflow)

2. (27 pts) Assume the same register contents contents as above. For EACH of the following two instruction sequences, tell if the jump is TAKEN or NOT TAKEN.

a. cmp dl,al
jle there **TAKEN** NOT_TAKEN
(signed, dl (negative) less than al (positive)

b. cmp dl,al
jae there **TAKEN** NOT_TAKEN
(unsigned, dl is higher than al)

c. cmp dl,al
je there TAKEN **NOT_TAKEN**
(dl not equal to al)

d. cmp dx,ax
jb there **TAKEN** NOT_TAKEN
(unsigned, dx is lower than ax)

e. cmp dx,ax
jg there **TAKEN** NOT_TAKEN
(signed, dx (positive) greater than ax (negative)

f. test al,1
jz there TAKEN **NOT_TAKEN**
(al AND 01 is nonzero, branch not taken)

g. add al,cl
jc there TAKEN **NOT_TAKEN**
(al + cl does not produce a carry, not taken)

h. add al,cl
jns there TAKEN **NOT_TAKEN**
(al + cl is a negative number MSB = 1, to NOT TAKEN)

i. add al, dl
jo there TAKEN **NOT_TAKEN**
(al + dl is positive + negative so no overflow)

3. (15 pts) For the address decoder below:

a. Give the range of addresses that output Y5 is valid for (Use HEX addresses)
340000 to 37FFF

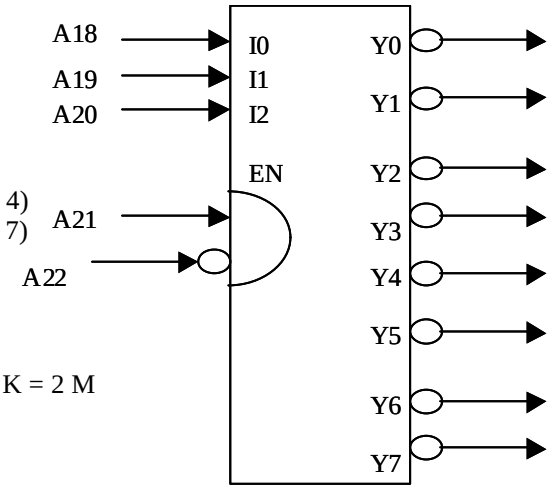
b.How many TOTAL bytes is this address decoder valid for? (give the answer in Kbytes, or Mbytes)

A22	A21	A20	A19	A18	A17-A0
0	1	1	0	1	0--0
0	1	1	0	1	1--1

Low 34 0000 (A23-A20 = 3, A18-A16= 4)
High 37 0000 (A23-A20 = 3, A18-A16= 7)

A17 thru A0 is 18 address lines
Y5 valid for $2^{18} = 2^8 * 2^{10} = 256K$

Total address Range is 8 * 256K = 2048 K = 2 M



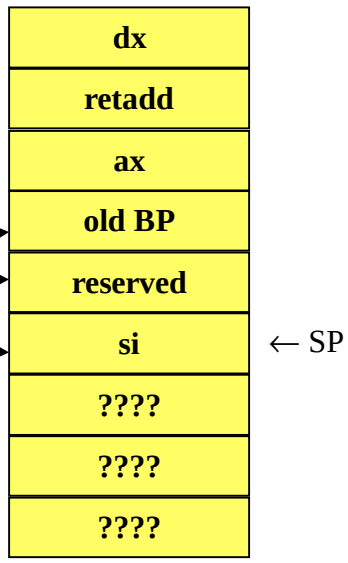
3-to-8 Decoder
I2 is MSB, I0 is LSB

4. For the following code:

```
                push dx
                call suba        ;;calling code
                ….               ;; other instructions
                ….

Suba            proc
                Push ax
                Enter  2,0
                Push si
                …..
                ;; some code here that modifies ax, si
                ;;
                ;;
                ????
                ????     ;; one or more instructions needed here…

suba            endp
```



a. (15 pts) Draw a stack picture that shows the state of the stack after the *'push s*i' instruction inside of *suba*. Draw the stack diagram as being 16 bits wide. Show all know registers/values that are on the stack, and show the position of BOTH the stack pointer and base pointer.

b. (10 pts) What code is needed at the end of the subroutine in order to return to the calling code and to clean the stack of ALL PASSED PARAMETERS (upon return to the calling code, the value of the SP should be restored to the value BEFORE the 'push dx' instruction. Also, registers 'ax' and 'si' need to be restored their original values before 'Suba' returns.

Do reverse of stack operations:

```
Pop si
Leave
Pop ax
Ret  2          (return and increment SP by 2 to get rid of space used by 'push dx')
```