EECS 388 HW #3

Due: March 4

Problems from textbook

Q1: Page 70: Challenging – 7 (20 points)
Write a program segment starting at $C100 that checks bits 0 and 2 of address $D000 and jumps to $C0CC if both bits are clear.

This task is easily accomplished using the BRCLR instruction. We are interested in bits 0 and 2, which gives us the 8-bit mask 00000101 ($05). Therefore:

```
        ORG     $C100
        LDX     #$D000
        BRCLR   0,X,$05,$C0CC
        SWI
```

Q2: Page 116: Advanced – 3 (25 points)
Write a program using a subroutine to copy a table from one location to another. A partially completed program is given next. Write a program by filling in locations where only comments appear.

Note 1: when we save the CPU registers, we have to make sure to restore them in reverse order to align the values properly and to make sure they go back to their proper places.

Note 2: FDB and FCB are not identical to EQU; when you load the values into a register, you do not use the immediate operator '#'.

```
* Copying a table using a subroutine
  Data Section
        ORG     $0000
TAB1    FDB     $D100       ; address of the first table
TAB2    FDB     $D300       ; address of the second table
TABL    FCB     $FF         ; table length

* main program
        ORG     $C100
        LDS     #$8000      ; initialize the stack pointer
        LDAA    TABL        ; load the table length to acc A
        LDX     TAB1        ; load table 1 address to X
        LDY     TAB2        ; load table 2 address to Y
        JSR     COPYT       ; call the subroutine
        SWI                 ; stop subroutine
        ORG     $4500
```

```
COPYT   PSHD                    ; save the CPU registers onto the stack
        PSHX
        PSHY
        PSHC
AGAIN   TSTA                    ; check the counter value
        BEQ     DONE            ; if zero jump to the end
        LDAB    1,X+            ; note the use of accumulator B
        STAB    1,Y+
        SUBA    #$01            ; adjust the counter and target addresses
        BRA     AGAIN           ; continue the loop
DONE    PULC                    ; restore the CPU registers
        PULY
        PULX
        PULD
        RTS                     ; IMPORTANT!!!!!!!!
        END
```

Q3: Page 116: Advanced - 4 (20 points)
Suppose you started with the following register contents.

P-C007 Y-7892 X-FF00 A-44 B-70 SP-C04A

What address is in the stack pointer and exactly what is in the stack after the following instruction sequence is executed?

```
PSHA
PSHB
PSHY
```

After the execution of the above instructions, the stack pointer points to address $C046 and the stack contains

Stack Pointer →

Address	Content
C045	…
C046	78
C047	92
C048	70
C049	44

As you can see, accumulator A is at the bottom of the stack, at address $C049, because it was pushed first. It is followed by accumulator B, and then register Y.

Q4: Page 117, Challenging - 1 (25 points)

Write a subroutine to copy data one byte at a time from memory location $5000 to memory location $6000 until a byte with $FF is detected.

Instead of just writing the subroutine, let us write the whole program to show what it would look like. This allows us to practice with manipulating the stack by saving and restoring registers.

```
TAB1    EQU     $5000       ; address of the first table
TAB2    EQU     $6000       ; address of the second table
EOS     EQU     $FF         ; end of string

        ORG     $4000
        LDS     #$8000      ; initialize the stack pointer
        LDX     #TAB1; load table 1 address to X
        LDY     #TAB2; load table 2 address to Y
        JSR     COPYT       ; call the subroutine
        SWI                 ; stop subroutine

COPYT   PSHD                ; save the CPU registers onto the stack
        PSHX
        PSHY

AGAIN   LDAA    1,X+        ; load byte into A and increment X
        CMPA    #EOS        ; compare value in A to EOS
        BEQ     DONE        ; if zero jump to the end
        STAA    1,Y+        ; store byte to table 2 and increment Y
        BRA     AGAIN       ; continue the loop

DONE    PULY                ; restore the CPU registers
        PULX
        PULD
        RTS                 ; return from subroutine
```

Q5: Page 117: Challenging - 8 (10 points)
Write an instruction sequence to load the contents of the element in the top of the stack onto accumulator A and the third element from the top of the stack onto accumulator B.

There is a similar example in the book on page 81. The following instructions do the job:

```
LDS     #$8000      ; initialize the stack pointer
TSX                 ; store stack pointer contents to register X
LDAA    0,X         ; load top element into A
LDAB    2,X         ; load third element into B
```