Key to Final Exam S3
Computer Architecture

Duration: 1 hr. 30 min.

Exercise 1 (9 points)

All questions in this exercise are independent. **Except for the output registers, none of the data or address registers must be modified when the subroutine returns.** A string of characters always ends with a null character (the value zero). A blank character is either a space character or a tab character.

1. Write the **IsBlank** subroutine that determines if a character is blank (i.e. if it is a space or a tab character).
   Input : **D1.B** holds the ASCII code of the character to test.
   Output : If the character is blank, **D0.L** returns 0.
   If the character is not blank, **D0.L** returns 1.

   Tip: The ASCII code of the tab character is 9.

```
IsBlank             ; If the character is a space, go to blank.
                    cmpi.b  #' ',d1
                    beq     \blank

                    ; If the character is a tab, go to blank.
                    cmpi.b  #9,d1
                    beq     \blank

\not_blank          ; The character is not blank, return D0.L = 1.
                    moveq.l #1,d0
                    rts

\blank              ; The character is blank, return D0.L = 0.
                    moveq.l #0,d0
                    rts
```

2. Write the **BlankCount** subroutine that returns the number of blank characters in a string. To know if a character is blank, use the **IsBlank** subroutine.
   Input : **A0.L** points to a string of character.
   Output : **D0.L** returns the number of blank characters in the string.

   Tips:
   - Use **D2** as a blank-character counter (because **D0** is used by **IsBlank**).
   - Then, copy **D2** into **D0** before returning from the subroutine.

```
BlankCount          ; Save registers on the stack.
                    movem.l d1/d2/a0,-(a7)

                    ; Initialize the blank-character counter.
                    clr.l   d2

\loop               ; Load a character from the string into D1.B.
                    ; If the character is null, go to quit.
                    move.b  (a0)+,d1
                    beq     \quit

                    ; If the character is not blank, go to loop.
                    jsr     IsBlank
                    tst.l   d0
                    bne     \loop

                    ; Otherwise, increment the counter.
                    addq.l  #1,d2
                    bra     \loop

\quit               ; Number of blank characters -> D0.L
                    move.l  d2,d0

                    ; Restore registers from the stack and return from subroutine.
                    movem.l (a7)+,d1/d2/a0
                    rts
```

3. Write the **BlankToUnderscore** subroutine that converts the blank characters in a string into underscore characters. To know if a character is blank, use the **IsBlank** subroutine.
   Input : **A0.L** points to a string of characters.
   Output : The blank characters of the string are replaced by the « _ » character.

```
BlankToUnderscore   ; Save registers on the stack.
                    movem.l d0/d1/a0,-(a7)

\loop               ; Load a character from the string into D1.B.
                    ; If the character is null, go to quit.
                    move.b  (a0)+,d1
                    beq     \quit

                    ; If the character is not blank, go to loop.
                    jsr     IsBlank
                    tst.l   d0
                    bne     \loop

                    ; Otherwise, the blank character is replaced
                    ; by the underscore character.
                    move.b  #'_',-1(a0)
                    bra     \loop

\quit               ; Restore registers from the stack and return from subroutine.
                    movem.l (a7)+,d0/d1/a0
                    rts
```

Exercise 2 (4 points)

Complete the table shown on the answer sheet. Write down the new values of the registers (except the **PC**) and memory that are modified by the instructions. **Use the hexadecimal representation**. **Memory and registers are reset to their initial values for each instruction**.

Initial values:

```
D0 = $0004FFFD   A0 = $00005000   PC = $00006000
D1 = $FFFF000A   A1 = $00005008
D2 = $FFFFFFFE   A2 = $00005010

$005000   54 AF 18 B9 E7 21 48 C0
$005008   C9 10 11 C8 D4 36 1F 88
$005010   13 79 01 80 42 1A 2D 49
```

Exercise 3 (3 points)

Complete the table shown on the answer sheet. Give the result of the additions and the values of the **N**, **Z**, **V** and **C** flags.

Exercise 4 (4 points)

Let us consider the following program:

```
Main        move.l  #$44AA77FF,d7 ; $44AA77FF -> D7.L

next1       moveq.l #1,d1         ; $00000001 -> D1.L
            tst.w   d7            ; Set N and Z according to D7.W.
            bmi     next2         ; Branch if N = 1 (D7.W < 0).
            moveq.l #2,d1         ; Otherwise, $00000002 -> D1.L

next2       clr.l   d2            ; $00000000 -> D2.L
            move.w  #$1234,d0     ; $1234 -> D0.W (D0.B = $34)
loop2       addq.l  #1,d2         ; D2.L + 1 -> D2.L
            subq.b  #1,d0         ; D0.B - 1 -> D0.B ; Only D0.B is decremented.
            bne     loop2         ; Branch if Z = 0 (D0.B ≠ 0)

next3       clr.l   d3            ; $00000000 -> D3.L
            move.w  #$1234,d0     ; $1234 -> D0.W
loop3       addq.l  #1,d3         ; D3.L + 1 -> D3.L
            dbra    d0,loop3      ; DBRA = DBF ; D0.W - 1 -> D0.W
                                  ; Branch if D0.W ≠ -1 (D0.W ≠ $FFFF)

next4       moveq.l #1,d4         ; $00000001 -> D4.L
            cmp.b   #$70,d7       ; Compare D7.B to $70.
            blt     quit          ; Branch if D7.B < $70 (signed comparison).
            moveq.l #2,d4         ; Otherwise, $00000002 -> D4.L

quit        illegal
```

Complete the table shown on the answer sheet.

Last name: .... First name: .... Group: ....

ANSWER SHEET TO BE HANDED IN WITH THE SCRIPT

Exercise 2

Instruction	Memory	Register
Example	`$005000  54 AF 00 40 E7 21 48 C0`	A0 = `$00005004` A1 = `$0000500C`
Example	`$005008  C9 10 11 C8 D4 36 FF 88`	No change
`MOVE.B  -1(A2),-(A1)`	`$005000  54 AF 18 B9 E7 21 48 88`	A1 = `$00005007`
`MOVE.L  $500E,-1(A1,D0.W)`	`$005000  54 AF 18 B9 1F 88 13 79`	No change
`MOVE.L  #$500E,-8(A0,D1.W)`	`$005000  54 AF 00 00 50 0E 48 C0`	No change
`MOVE.W  $500A(PC),-(A1)`	`$005000  54 AF 18 B9 E7 21 11 C8`	A1 = `$00005006`

Exercise 3

Operation	Size (bits)	Result (hexadecimal)	N	Z	V	C
`$A3 + $5C`	8	`$FF`	1	0	0	0
`$7005 + $7005`	16	`$E00A`	1	0	1	0
`$7FFFFFFF + $80000001`	32	`$00000000`	0	1	0	1

Exercise 4

Values of registers after the execution of the program. **Use the 32-bit hexadecimal representation.**	
D1 = $00000002	**D3** = $00001235
D2 = $00000034	**D4** = $00000001