COEN 311 – Computer Organization and Software

Mid-Term Examination

Winter 2019

Student Identification

First Name	Surname	ID Number

Exercise 1

Give the result of the following set of 2's complement additions. Indicate if an overflow occurs.

Addition	Result	Overflow ?
1111 1101 + 0010 0011	0010 0000	No
0100 1001 + 0011 1000	1000 0001	Yes

EXERCISE 2

A and B are real numbers in single precision format (ieee-754 floating point). Give the hexadecimal representation of their ieee-754 floating point sum.

Item	Value
A	0x3FC00000
B	0x3FE00000
A + B	0x40500000

EXERCISE 3

The following table presents different cache memories. The address bus is 16 bits wide. Fill the table with appropriate values. Leave blank when field does not apply.

Size	Type	Nb Sets	Nb bytes per block	Tag Address Width	Set/Group Address Width	Byte Address Width
16 kB	Direct Mapped	N/A	16	2	10	4
16 kB	Set Associative	4-way	16	4	8	4

EXERCICE 4

Assume:

```
PC = $2DF8
SR = $0004
(a0, a1, a2, a3) = ($00000000, $00000002, $00000004, $FFFF0006)
(d0, d1, d2, d3) = ($00000002, $00000004, $00000006, $FFFF0007)
```

Provide the hex code for each of the following SIM68 instruction. Refer to the SIM68 Card provided in appendix

Instruction	Hex Code
`movea d2, a2`	$3442
`beq $2DF0`	$67F6
`sub d1, d0`	$9041
`adda d0, a3`	$D6C0

EXERCICE 5

Consider the following SIM68 assembly code.

```
01          org   $0000
02    _main sub   d0, d0
03          move  N(a0), d3
04          move  DECR(a0), d1
05          movea d3, a0
06          move  d0, d3
07    LOOP  move  VALS(a0), d2
08          cmp   d0, d2
09          beq   THEN
10          add   d2, d3
11    THEN  adda  d1, a0
12          bgt   LOOP
13          movea d0, a0
14          move  d3, SUM(a0)
15          move  EXIT(a0), d0
16          trap  #0
17    N     dc    20
18    DECR  dc    -6
19    VALS  dc    1, 2, 0, 4, 0, 6, 0, 8, 0, 10, 11
20    EXIT  dc    3
21    SUM   ds    1
22          end
```

What is the value taken by (`d0, d1, d2, d4`) when the program terminates?

Data Registers	Value in hex (low order word)
d0	$0003
d1	$FFFA
d2	$0000 ($0002)
d3	$0013 ($0015)

APPENDIX

Arithmetic Instructions

Add: Mnemonic **add** or **adda**.

Format	Bit Assignment					Length
F1	OP_4	drn_3	om_3	sMS_6		1
	1101	ddn_3	001	000	sdn_3	
	1101	dan_3	011	000	sdn_3	

Subtract: Mnemonic **sub**.

Format	Bit Assignment					Length
F1	OP_4	drn_3	om_3	sMS_6		1
	1001	ddn_3	001	000	sdn_3	

Multiplication: Mnemonic **muls**.

Format	Bit Assignment					Length
F3	OP_4	drn_3	OP_3	sMS_6		1
	1100	ddn_3	111	000	sdn_3	

Division: Mnemonic **divs**.

Format	Bit Assignment					Length
F3	OP_4	drn_3	OP_3	sMS_6		1
	1000	ddn_3	111	000	sdn_3	

Comparison: Mnemonic **cmp**.

Format	Bit Assignment						Length
F2	OP_4	ddn_3	OP_1	om_2	sMS_6		1
	1011	ddn_3	0	01	000	sdn_3	

Moving instructions

Swap: Mnemonic **swap**.

Format	Bit Assignment		Length
F7	OP_{13}	dn_3	**1**
	0100100001000	dn_3	

Register-to-Register Transfers: Mnemonic **move** or **movea**.

Format	Bit Assignment					Length
F4	OP_2	om_2	drn_3	dmd_3	sMS_6	1, 2, 3
	00	11	ddn_3	000	000 sdn_3	
	00	11	dan_3	001	000 sdn_3	
	00	11	ddn_3	000	001 san_3	
	00	11	dan_3	001	001 san_3	

Register-to-Memory Transfers: Mnemonic **move**.

Format	Bit Assignment						Length
F4	OP_2	om_2	drn_3	dmd_3	sMS_6		1, 2, 3
	00	11	dan_3	101	000	sdn_3	$displacement_{16}$
	00	11	dan_3	101	001	san_3	$displacement_{16}$

Memory-to-Register Transfers: Mnemonic **move** or **movea**.

Format	Bit Assignment						Length
F4	OP_2	om_2	drn_3	dmd_3	sMS_6		1, 2, 3
	00	11	ddn_3	000	101	san_3	$displacement_{16}$
	00	11	dan_3	001	101	san_3	$displacement_{16}$

Memory-to-Memory Transfers: Mnemonic **move**.

Format	Bit Assignment							Length
F4	OP_2	om_2	drn_3	dmd_3	sMS_6			1, 2, 3
	00	11	ddn_3	101	101	san_3	$sdisp_{16}$	$ddisp_{16}$

Branching Instructions

Unconditional branch: Mnemonic **bra**.

Format	Bit Assignment			Length
F5	OP_4	$Cond_4$	$Displacement_8$	1, 2
	0110	0000	$displacement_8$	
	0110	0000	00000000	$displacement_{16}$

Branch on equal: Mnemonic **beq**, branches if (Z=1).

Format	Bit Assignment			Length
F5	OP_4	$Cond_4$	$Displacement_8$	1, 2
	0110	0111	$displacement_8$	
	0110	0111	00000000	$displacement_{16}$

Branch on greater than: Mnemonic **bgt**, branches if (Z, N, O) = (0, 0, 0) or (0, 1, 1).

Format	Bit Assignment			Length
F5	OP_4	$Cond_4$	$Displacement_8$	1, 2
	0110	1110	$displacement_8$	
	0110	1110	00000000	$displacement_{16}$

Branch on less than: Mnemonic **blt**; branches if (N, O) = (0, 1) or (1, 0).

Format	Bit Assignment			Length
F5	OP_4	$Cond_4$	$Displacement_8$	1, 2
	0110	1101	$displacement_8$	
	0110	1101	00000000	$displacement_{16}$

Stop and Dump: Mnemonic **trap**.

Format	Bit Assignment		Length
F6	OP_{12}	V_4	1
	010011100100	0000	

$d0_{16}$ = {0, 1, 2, 3} eq. {mem and reg, mem, reg, stop}