Name-surname : **Signature :**

Student no : **Email :**

CSE2201(2204) - Computer Organization (Final Exam (21.01.2008))

Answer either 1a *or* 1b

1a. (a) Write and explain the structural components of any general-purpose computer? *(5 pts)*
(b) Write the functions of control unit. *(5 pts)*

1b. (a) What types of transfers must a computer's interconnection structure (or bus) support? *(5 pts)*
(b) Write the memory hierarchy in a computer system. *(5 pts)*

Answer either 2a *or* 2b

2a. Convert the given hexadecimal number $(\mathbf{12.C})_{16}$ to the corresponding 32 bit IEEE floating point number. *(10 pts)*

2b. If $\mathbf{X=(A)_{16}}$ and $\mathbf{Y=(4)_{16}}$, then find **X–Y** and comment on the results. Assume twos complement numbers. *(10 pts)*

3. In a processor, arithmetic operations are implemented in 2s complement notation. If given numbers are **C** and **4** in hexadecimal, compute the product of these numbers using Boots algorithm? *(15 pts)*

4. Consider a 16 bit processor in which the following appears in main memory, starting at location 200:

200	Load to AC	Mode
201	500	
202	Next instruction	

The first part of the first word indicates that this instruction loads a value into an accumulator (AC). The Mode field specifies an addressing mode and, if appropriate, indicates a source register; assume that when used, the source register is **R1**, which has a value of **400**. The value of **500** in location **201** may be part of the address calculation. Assume that location **399** contains the value **999**, location **400** contains the value **1000**, location **401** contains the value **1001**, and so on. Determine the effective address and the operand to be loaded for the following addressing modes:

(a) Direct *(4 pts)*
(b) Immediate *(4 pts)*
(c) Indirect *(4 pts)*
(d) Register *(4 pts)*
(e) Register indirect *(4 pts)*

5. If one line of an assembler program for a hypothetical processor and corresponding machine code is given as:

address	assembler	machine code	
20000	lda A4F05	12A4F05	(the values are in hexadecimal)

(a) How many instructions this processor might have? *(4 pts)*
(b) How many bits are needed for the instruction register? *(4 pts)*
(c) Assuming that bytewise addressing is used, how many memory location an instruction will occupy? *(4 pts)*
(d) Assuming that bytewise addressing is used, what is the total directly addressable memory size ? *(4 pts)*
(e) For the memory location A4F05, find the Tag, Line, and Word values in hexadecimal format for a direct-mapped cache, when tag-id=5 bits, line-id=8 bits, word-id=7 bits. *(4 pts)*

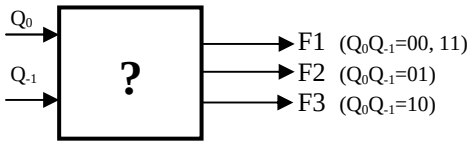
6.

```
00   in       // input number A
01   sta 20
02   in       // input number B
03   sta 21
04   sub 20
05   brp 08
06   lda 20
07   br  10
08   brz 10
09   lda 21
10   sta 50
11   hlt
```

What will be the content of the memory location **50** after running the given VVM code when inputs entered as:

(a) A = 5, B = 3 *(4 pts)*
(b) A = 2, B = 6 *(4 pts)*
(c) A = 4, B = 4 *(4 pts)*
(d) What does this code do? Explain *(4 pts)*

7. In Booth algorithm, Q_0 and Q_{-1} are checked at each step to determine one of the three necessary actions:
(AShift through A,Q, and Q_{-1}; A ← A+ M, AShift through A,Q, and Q_{-1}; A ← A - M, AShift through A,Q, and Q_{-1}).
Design the required logic circuit. *(10 pts)*



VVM instruction set

Load Accumulator (5*nn*) [LDA *nn*]
The content of RAM address *nn* is copied to the Accumulator Register, replacing the current content of the register. The content of RAM address *nn* remains unchanged. The Program Counter Register is incremented by one.
Store Accumulator (3*nn*) [STO *nn*] or **[STA *nn*]**
The content of the Accumulator Register is copied to RAM address nn, replacing the current content of the address. The content of the Accumulator Register remains unchanged. The Program Counter Register is incremented by one.
Add (1*nn*) [ADD *nn*]
The content of RAM address nn is added to the content of the Accumulator Register, replacing the current content of the register. The content of RAM address nn remains unchanged. The Program Counter Register is incremented by one.
Subtract (2*nn*) [SUB *nn*]
The content of RAM address nn is subtracted from the content of the Accumulator Register, replacing the current content of the register. The content of RAM address nn remains unchanged. The Program Counter Register is incremented by one.
Input (901) [IN] or **[INP]**
A value input by the user is stored in the Accumulator Register, replacing the current content of the register. Note that the two-digit operand does not represent an address in this instruction, but rather specifies the particulars of the I/O operation (see Output). The operand value can be omitted in the Assembly Language format. The Program Counter Register is incremented by one with this instruction.
Output (902) [OUT] or **[PRN]**
The content of the Accumulator Register is output to the user. The current content of the register remains unchanged. Note that the two-digit operand does not represent an address in this instruction, but rather specifies the particulars of the I/O operation (see Input). The operand value can be omitted in the Assembly Language format. The Program Counter Register is incremented by one with this instruction.
Branch if Zero (7*nn*) [BRZ *nn*]
This is a conditional branch instruction. If the value in the Accumulator Register is zero, then the current value of the Program Counter Register is replaced by the operand value nn (the result is that the next instruction to be executed will be taken from address nn rather than from the next sequential address). Otherwise (Accumulator >< 0), the Program Counter Register is incremented by one (thus the next instruction to be executed will be taken from the next sequential address).
Branch if Positive or Zero (8*nn*) [BRP *nn*]
This is a conditional branch instruction. If the value in the Accumulator Register is nonnegative (i.e., >= 0), then the current value of the Program Counter Register is replaced by the operand value nn (the result is that the next instruction to be executed will be taken from address nn rather than from the next sequential address). Otherwise (Accumulator < 0), the Program Counter Register is incremented by one (thus the next instruction to be executed will be taken from the next sequential address).
Branch (6*nn*) [BR *nn*] or**[BRU *nn*]** or **[JMP *nn*]**
This is an unconditional branch instruction. The current value of the Program Counter Register is replaced by the operand value nn. The result is that the next instruction to be executed will be taken from address nn rather than from the next sequential address. The value of the Program Counter Register is not incremented with this instruction.
No Operation (4*nn*) [NOP] or **[NUL]**
This instruction does nothing other than increment the Program Counter Register by one. The operand value nn is ignored in this instruction and can be omitted in the Assembly Language format. (This instruction is unique to the VVM and is not part of the Little Man Model.)
Halt (0*nn*) [HLT] or **[COB]**
Program execution is terminated. The operand value nn is ignored in this instruction and can be omitted in the Assembly Language format.

By default, programs are loaded in sequential addresses starting at 00. Memory load address can be adjusted with the load directive "*nn", where "nn" is a two-digit address. When this directive is found, subsequent statements are loaded beginning with that new address. Prefixing a program statement with "nn:" will also adjust the load address of that and subsequent statements.

Data values used by a program can be loaded into memory along with the program. In Machine or Assembly Language form simply use the format "snnn" where s is an optional sign, and nnn is the three-digit data value. In Assembly Language, you can specify "DAT snnn" for clarity.

1a. (a) **CPU**: for processing data
Memory: For storing data
Input/Output: For moving data in and out of computer
System interconnection: For moving data amongst other components
(b) **Sequencing:** The control unit causes the processor to step through a series of micro-operations in the proper sequence, based on the program being executed.
Execution: The control unit causes each micro-operation to be performed.

1b. (a) The interconnection structure must support the following types of transfers:
Memory to processor
Processor to memory
I/O to processor
Processor to I/O
I/O to or from memory

(b) Registers, L1 Cache, L2 Cache, Main memory, Disk cache, Disk, Optical, Tape

---

2a. $(12.C)_{16} = (00010010.1100)_2$
$(12.C)_{16} = (10010.1100)_2 \Rightarrow (1.001011\times2^4) = (1.001011\times2^{00000100})$
biased exponential format = $(1.001011\times2^{4+127}) = (1.001011\times2^{00000100+01111111})$
biased exponential format = $(1.001011\times2^{10000011})$

S = 0, BE = 10000011, M = 00101100000000000000000

$(12.C)_{16}$=(0 10000011 00101100000000000000000)$_{\text{floating point format}}$

2b. (a) $X = (A)_{16} = (1010)_2$, $Y = (4)_{16} = (0100)_2$
The most significant bit of X is 1, so X is a negative number. The most significant bit of Y is 0, so Y is a positive number.

			0 (C_{in})	
X	1 0 1 0		1 0 1 0	
-Y	-0 1 0 0	➔	+1 1 0 0	(2s complement of 0100)
		(C_{out}) 1	0 1 1 0	

Because C_{in} and C_{out} are different, there is an overflow and the result is incorrect.
Four bits are too small to hold the correct answer.

---

3. If $Q = (4)_{16} = (0100)_2 = (4)_{10}$ and $M = (C)_{16} = (1100)_2 = (-4)_{10}$, then the solution:

	A	Q	Q_{-1}	M	Process
	0000	0100	0	1100	Initial
Step 1	0000	0010	0	1100	AShift
Step 2	0000	0001	0	1100	AShift
Step 3	0100	0001	0	1100	A ← A – M
	0010	0000	1	1100	AShift
Step 4	1110	0000	1	1100	A ← A + M
	1111	0000	1	1100	AShift

If $Q = (C)_{16} = (1100)_2 = (-4)_{10}$ and $M = (4)_{16} = (0100)_2 = (4)_{10}$, then the solution:

	A	Q	Q_{-1}	M	Process
	0000	1100	0	0100	Initial
Step 1	0000	0110	0	0100	AShift
Step 2	0000	0011	0	0100	AShift
Step 3	1100	0011	0	0100	A ← A – M
	1110	0001	1	0100	AShift
	1111	0000	1	0100	AShift

$(11110000)_2$ (in 2s complemenet notation) = $(-16)_{10}$

4.

	Effective Address	Operand
(a)	500	1100
(b)	201	500
(c)	1100	1700
(d)	R1	400
(e)	400	1000

5.

(a) Op code field is 8 bits. So this processor might have up to $2^8 = 256$
(b) 28 bits.
(c) Instruction length is 3.5 byte. So it needs at least 4 memory locations
(d) Address length is 5 nibbles (5*4=20 bits). So, total directly addressable memory is 2^{20}=1Mbyte.

(e) First, write the address in binary format:

$(A4F05)_{16} = (1010\ 0100\ 1111\ 0000\ 0101)_2$

Then define the Tag, Line, and Word values:

Tag-id← 10100<u>10011110</u>0000101 →Word-id
Line-id

Tag-id = 00010100 = $(14)_{16}$

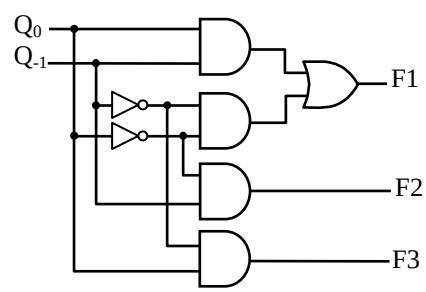
Line-id = <u>10011110</u> = $(9E)_{16}$

Word-id = 00000101 = $(05)_{16}$

---

6. (a) A = 5 will be stored in memory location 50.
(b) B = 6 will be stored in memory location 50.
(c) 0 will be stored in memory location 50.
(d) This program compares two numbers (A and B) entered from the keyboard. As a result of the comparison A, B, or 0 is stored in memory location 50 depending on the following conditions:
If A > B, A is stored in memory location 50
If A < B, B is stored in memory location 50
If A = B, 0 is stored in memory location 50

7.

Q_0	Q_{-1}	F1	F2	F3
0	0	1	0	0
0	1	0	1	0
1	0	0	0	1
1	1	1	0	0

$$F1 = \overline{Q_0}\,\overline{Q_{-1}} + Q_0 Q_{-1}$$
$$F2 = \overline{Q_0} Q_{-1}$$
$$F3 = Q_0 \overline{Q_{-1}}$$