Name-surname : **Signature :**
Student no : **Email :**

1st Midterm Exam (16.11.2007)

CSE2201(2204)- Computer Organization

1. List the main components and their functions of today's stored-program computers? *(20 pts)*

2. A computer system has an 8 bit accumulator and 12 bit address bus. If the content of memory location 1F0 is the binary number 11000011;
   (a) what is the corresponding decimal value when the number represents an unsigned integer number? *(5 pts)*
   (b) what is the corresponding decimal value when the number represents a twos complement number? *(5 pts)*
   (c) what is the corresponding decimal value when the number represents a ones complement number? *(5 pts)*
   (d) Assuming that the most significant bit is parity bit, what is the corresponding letter when the number represents an ASCII character (hint: ASCII character is represented by 7 bits and 41H represents A)? *(5 pts)*

3. If the processor described in question 2 has maximum 16 instructions:
   (a) How many bits are needed for the op-code field of the instruction? *(5 pts)*
   (b) How many bits are needed for the instruction register? *(5 pts)*
   (c) What is the total memory size in kByte? *(5 pts)*
   (d) For the memory location 1F0, find the Tag, Set, and Word values in hexadecimal format for a set-associative cache, when tag-id=4 bits, set-id=6 bits, word-id=2 bits. *(5 pts)*

4. In a microprocessor, each machine instruction cycle requires 4 clock cycles to be completed.
   (a) If the clock frequency of the processor is 2 GHz, what will be the time required to complete an instruction cycle? *(10 pts)*
   (b) What would be the duration of a program consisting of 200 instructions? *(10 pts)*

5. Write a simple VVM Assembly program that if the number entered from the keyboard is negative, it will be converted to pozitive and saved in the memory location 20, if positive the data will be just stored in the memory location 20. For the temporary storage use any memory location between 30 and 39. *(20 pts)*

Turn the page for the VVM instruction set ➔

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.

1. **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

---

2. binary 11000011:
   (a) 195
   (b) -61
   (c) -60
   (d) If the most significant bit is parity bit, ASCII character represented by 11000011 is 100 0011 (43 in hexadecimal) corresponding to capital letter ***C***.

3.
   a. If the processor has 16 instructions, $\log_2 16 = 4$ bits are needed for op-code field
   b. Instruction is *op-code+operand* (*address*). Addres is 12 bits. IR size = 4 + 12 = 16 bits
   c. Total memory size = $2^{12} = 2^{10+2} = 2^2 \times 2^{10}$ = 4 kByte
   d. First, write the address in binary format: $(1F0)_{16} = (0001\ 1111\ 0000)_2$

   Then define the Tag, Set, and Word values:

   Tag-id← 0001<u>11110</u>00 →Word-id

   Set-id

   Tag-id = 0001 = 1; Set-id = <u>0011</u>1100 = 3C; Word-id = 0000 = 0

4.
   (a) Clock period = 1 / clock frequency = 1/ 2 GHz = 0.5×10^9 s = 0.5 ns.
   Time required to complete an instruction cycle = 0.5× 4 = 2 ns.
   (b) Duration of a program consisting of 200 instructions = 2× 200 = 400 ns

---

5. One solution is as follows:

```
00   in        // input number
01   brp 05    // if +ve jump to 05
02   sto 30    // else store in 30
03   sub 30    // -x – (-x) = 0
04   sub 30    // 0 – (-x) = x
05   sto 20    // save the number
06   hlt       // stop
```