NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING

EXAMINATION FOR

Semester 1 AY2008/9

CS2100 – COMPUTER ORGANISATION

Nov 2008 **Time allowed: 2 hours**

Your Matriculation Number:

INSTRUCTIONS TO CANDIDATES

1. This examination paper consists of **EIGHT (8)** questions and comprises **NINETEEN (19)** printed pages including this page.
2. This is an **OPEN BOOK** examination. You may use any approved calculators but not any PDA or laptop, especially those capable of external connectivity or communication.
3. Answer *all* questions. Note that the full mark for each question is different.
4. Write your answers on *this* **QUESTION AND ANSWER SCRIPT**. Answer only in the space given. Any writing outside this space will not be considered. No other submission is allowed.
5. Fill in your Matriculation Number with a pen, clearly on every page of this QUESTION AND ANSWER SCRIPT.
6. You may use pencil to write your answers.
7. At the end of the examination, please check to ensure that your script has all the pages properly stapled together.
8. Note that when a number is written as "0xNNNN" it means that "NNNN" is in base 16.

Total Score _____ /100

QUESTION 1 (10 marks)
The following is part of the content of memory:

Memory Location (in hexadecimal)	Content of word (in hexadecimal)
0x00000200	0x300
0x00000204	0x200
0x00000208	0x100
0x0000020c	0x1
0x00000210	0x0
. . .	. . .
0x00400000	0x200d0200
0x00400004	0x8db00004
0x00400008	0x8e040000
0x0040000c	0x008d1020
0x00400010	0xae020000

And the registers contain the following:

Values in the register file (all in hexadecimal)

```
R0  (r0) = 00000000 | R1  (at) = 00002000
R2  (v0) = 00000001 | R3  (v1) = 0000000a
R4  (a0) = 00000005 | R5  (a1) = 7ffff000
R6  (a2) = 7ffff004 | R7  (a3) = 000000b0
R8  (t0) = 00000001 | R9  (t1) = 00000c00
R10 (t2) = 0000c000 | R11 (t3) = fffffff0
R12 (t4) = f0000000 | R13 (t5) = 00000fff
R14 (t6) = 00001000 | R15 (t7) = 00000e00
R16 (s0) = 00300000 | R17 (s1) = 00000c00
R18 (s2) = 00040200 | R19 (s3) = 00011000
R20 (s4) = 00030200 | R21 (s5) = 10000000
R22 (s6) = 00055000 | R23 (s7) = f0000000
R24 (t8) = 00000005 | R25 (t9) = 0000d000
R26 (k0) = 00000000 | R27 (k1) = 00000000
R28 (gp) = 10008000 | R29 (sp) = 7fffeffc
R30 (s8) = 1000000f | R31 (ra) = 00400018
```

The PC is pointing at location 0x00400000.

(1a) What are the five MIPS instructions starting from location `0x00400000`? (5 marks)

ANSWER:

(1b) At the end of the execution of the fifth instruction starting from location `0x00400000` what are the contents of memory and the registers that have changed? (5 marks)

ANSWER:

QUESTION 2 (10 marks)
The `bgt` (branch if greater) and `bge` (branch if greater or equal) assembly instructions are pseudo-instructions. Show:

(2a) How is

```
bge $t0, $t1, L1
```

implemented using real MIPS instructions (instructions with actual opcodes).
(5 marks)

ANSWER:

(2b) How is

```
bgt $t0, $t1, L1
```

implemented using real MIPS instructions (instructions with actual opcodes).
(5 marks)

ANSWER:

QUESTION 3 (20 marks)

The C snippet below counts the number of blanks (ASCII value 32) and white tabs (ASCII value 9) in a C string. A C string is an array of bytes that is terminated by a last element that is zero. All other elements in this byte array are non-zero.

```
count = 0;
index = 0;
while (str[index] != 0) {
      if ((str[index] == 32)     /* blank = 32 */
          || (str[index] == 9)) /* tab = 9 */
         count++;
      index++;
}
```

Assume that `count` is in `$t0` and `index` is in `$t1` while the address of `str` is in `$s0`. Convert the above C code snippet into

(3a) `if-goto` pseudo-code. (10 marks)

ANSWER:

(3b) MIPS assembly code by assuming that the starting address of the array `str` is in `$s0` and the final count is to be left in `$v0`. (10 marks)

ANSWER:

QUESTION 4 (15 marks)
Consider the process of adding two IEEE Standard 754 single precision numbers A = `0x3d51265f` and B = `0x3c4ef353` in hexadecimal.

(4a) What is the amount of denormalization shifts needed to align the binary points? (3 marks)

ANSWER:

(4b) Perform the shift of the smaller number using the 3 bits (i.e. guard, round and sticky bits). Leave your answer in base 16. (3 marks)

ANSWER:

(4c) Perform the addition of the mantissa parts of the two numbers (one of which has been denormalized by the shifting in Part (b)) using the 3 bits of guard, round and sticky bits, leaving the answer in base 16. (3 marks)

ANSWER:

(4d) What is the direction (left or right) and amount of the shifts needed to normalize the result? (3 marks)

ANSWER:

(4e) What is the final result of the addition in base 16 after round-to-nearest is performed? (3 marks)

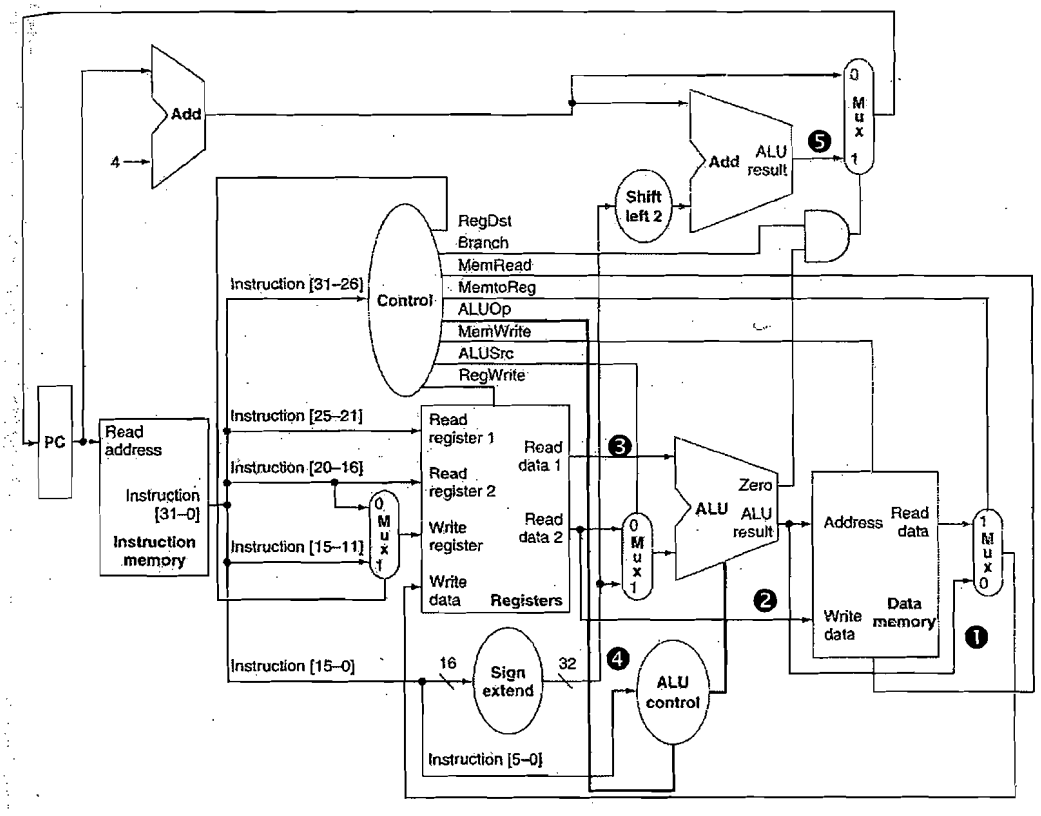
ANSWER:

QUESTION 5 (8 marks)

Consider the single cycle datapath with the values of the registers shown below.

Values in the register file (all in base 16)

R0	(r0)	= 00000000	R1	(at)	= 00002000
R2	(v0)	= 00000001	R3	(v1)	= 0000000a
R4	(a0)	= 00000005	R5	(a1)	= 7ffff000
R6	(a2)	= 7ffff004	R7	(a3)	= 000000b0
R8	(t0)	= 00000001	R9	(t1)	= 00000c00
R10	(t2)	= 0000c000	R11	(t3)	= fffffff0
R12	(t4)	= f0000000	R13	(t5)	= 00000fff
R14	(t6)	= 00001000	R15	(t7)	= 00000e00
R16	(s0)	= 00300000	R17	(s1)	= 00000c00
R18	(s2)	= 00040200	R19	(s3)	= 00011000
R20	(s4)	= 00030200	R21	(s5)	= 10000000
R22	(s6)	= 00055000	R23	(s7)	= f0000000
R24	(t8)	= 00000005	R25	(t9)	= 0000d000
R26	(k0)	= 00000000	R27	(k1)	= 00000000
R28	(gp)	= 10008000	R29	(sp)	= 7fffeffc
R30	(s8)	= 1000000f	R31	(ra)	= 00400018



The current PC is 200 and the instruction being executed is:

```
lw $t2, -4($sp)
```

Fill in the values of the fields (in hexadecimal) in the table below:

ANSWER:

Field	Value (in hexadecimal)
RegDst	
MemRead	
Branch	
MemtoReg	
MemWrite	
ALUSrc	
RegWrite	
Instruction[31-26]	
Instruction[25-21]	
Instruction[20-16]	
Instruction[15-11]	
❶	
❷	
❸	
❹	
❺	

QUESTION 6 (12 marks)
A file consists of 8 file blocks, each 1024 bytes. The file contains a running sequence of 4-byte binary words starting from 0 in big endian form, i.e.

File byte position	Byte content
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	0
9	0
10	0
11	2
.	.
.	.
.	.

Using the notation "blk[*i*]" to stand for the *i*-th block, "blk[*i*].byte[*j*]" to mean "byte *j* of block *i*" and "blk[*i*].byte[*j*].bit[*k*]" to stand for "bit *k* of byte *j* of block *i*", show how this file would be stored on:

(6a) A RAID Level 0 system consisting of 4 disks. (4 marks)

ANSWER:

Disk 0: **Disk 1:**

Disk 2: **Disk 3:**

(6b) A RAID Level 1 system consisting of 2 disks. (4 marks)

ANSWER:

Disk 0:

Disk 1:

(6c) A RAID Level 0+1 system consisting of 4 disks. (4 marks)

ANSWER:

Disk 0: **Disk 1:**

Disk 2: **Disk 3:**

QUESTION 7 (15 marks)
A machine with byte addresses and a word size of 32 bits and address width of 32 bits has a direct-mapped cache with 4 blocks each consisting of 2 words.

(7a) Given the MIPS program below, and assuming that array `A` starts at memory hexadecimal location `0x1000` while array `B` starts at memory hexadecimal location `0x4010`. Fill in the first 10 address requests seen at the data cache and indicate whether the reference is a hit (H) or a miss (M). Assume that the cache is initially empty. (5 marks)

```
        la   $s0, A
        la   $s1, B
        li   $t0, 1
loop:
        slt  $t1, $t0, 1000
        beq  $t1, $zero, end_loop
        sll  $t2, $t0, 2
        add  $t3, $s0, $t2
        lw   $a0, 0($t3)
        add  $t4, $s1, $t2
        lw   $a1, 0($t4)
        add  $v0, $a0, $a1
        sw   $v0, -4($t3)
        addi $t0, $t0, 1
        j    loop
end_loop:
        .    .    .
```

ANSWER:

Memory address seen at DS (in hexadecimal)	Hit (H) or Miss (M)?

(7b) Given the above program, fill in the final contents of the cache. Use the notation M[*i*] to denote the word at memory address *i*. (*i* may be in hexadecimal.) (6 marks)

ANSWER:

Index	Tag	Block 0	Block 1
0			
1			
2			
3			

(7c) What is the total number of data cache memory references, hits and misses after the execution of the above MIPS program? (4 marks)

ANSWER:

QUESTION 8 (10 marks)
A machine has a physical address width of 32 bits and a virtual address width of 32 bits. A page is 8192 bytes.

(8a) What is the maximum size of the virtual address space? (2 marks)

ANSWER:

(8b) What is the maximum size of the physical address space? (2 marks)

ANSWER:

(8c) Given the following content of a full associative 4-entry TLB:

Valid Bit	Virtual Page Number	Physical Page Number	Dirty	Ref	Access
1	0x20531	0x102c8	1	20	rw
1	0x814c5	0x84c5	0	1	rwx
0	0x40A6A	0x40AA	0	8	rw
1	0x40A62	0x5A62	1	9	rw

What is the physical address for the virtual address 0x40A62E90? (3 marks)

ANSWER:

(8d) Given that the bit length of the physical and virtual addresses is the same, give two reasons why virtual addressing is still desirable. (3 marks)

ANSWER:

=== END OF PAPER ===