NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING

EXAMINATION FOR
Semester 2 AY2011/2012

CS2100 – COMPUTER ORGANISATION

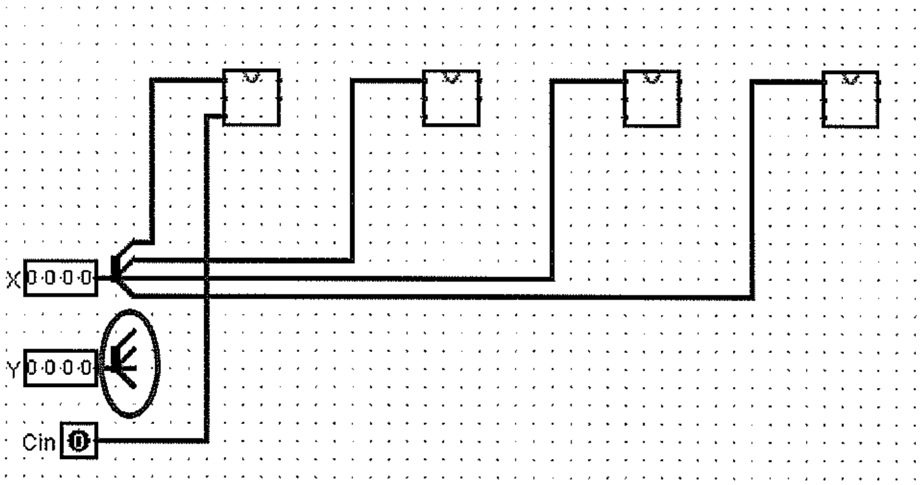
April 2012 **Time allowed: 2 hours**

INSTRUCTIONS TO CANDIDATES

1. This examination paper consists of **THIRTEEN (13)** questions and comprises **ELEVEN (11)** printed pages.
2. This is a **CLOSED BOOK** examination. Two handwritten A4 reference sheets are allowed. Calculators are not allowed.
3. Answer all questions.
4. Write your answers in the **ANSWER BOOKLET** provided.
5. Fill in your Matriculation Number with a pen, clearly on odd-numbered pages of your ANSWER BOOKLET.
6. You may use pencil to write your answers.
7. You are to submit only the **ANSWER BOOKLET** and no other document.

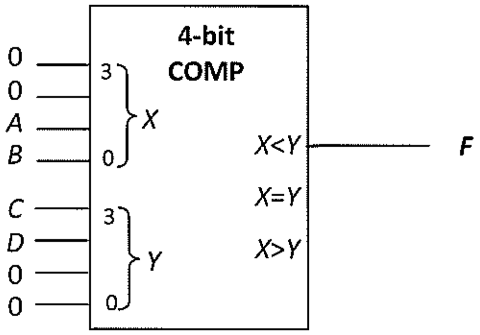
Questions 1 - 8: Each question has only one correct answer. Write your answers in the boxes provided in the Answer Booklet. One mark is awarded for a correct answer and no penalty for wrong answer.

1. The following diagram of a partially constructed circuit is taken from logisim. What is the circled wiring element called?



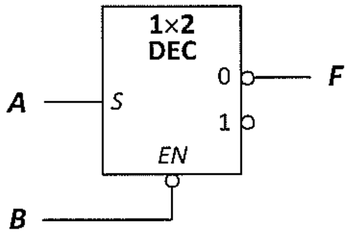
   A. Pin
   B. Splitter
   C. Bit Selector
   D. Label
   E. None of the above

2. Given the following circuit using a 4-bit magnitude comparator, what is the function ***F(A,B,C,D)***?



   A. $\Pi M(4, 8, 12)$
   B. $\Pi M(0, 4, 8, 12)$
   C. $\Sigma m(4, 8, 12)$
   D. $\Sigma m(0, 4, 8, 12)$
   E. None of the above

3. Given the following circuit using a 1×2 decoder with zero-enable and negated outputs, what is the function ***F(A,B)***?



A. $\Sigma m(0, 1, 2)$
B. $\Sigma m(0, 1, 3)$
C. $\Sigma m(0, 2, 3)$
D. $\Sigma m(1, 2, 3)$
E. None of the above

4. A decimal code is a code that represents the ten decimal digits 0 through 9. A certain converter converts a 4-bit decimal code *ABCD* into its equivalent 4-bit excess-6 code *WXYZ*. The simplified expressions for its outputs *W*, *X*, *Y* and *Z* are given below, considering the don't-care values.

$$W = A + B \cdot C' + C \cdot D'$$

$$X = B' \cdot C' + B' \cdot D' + B \cdot C \cdot D$$

$$Y = C' \cdot D' + C \cdot D$$

$$Z = D$$

What converter is this?

A. An 8421 code to excess-6 code converter.
B. An 84-2-1 code to excess-6 code converter.
C. A 2421 code to excess-6 code converter.
D. An excess-3 code to excess-6 code converter
E. None of the above

5. Due to inexperience, Mr. Blunder made a serious error in the 5-stage MIPS pipeline processor design. The register file is incorrectly implemented, such that in every cycle, the register read access is **performed before** the register write access. Which of the following statements regarding the processor behaviour is **TRUE**?

A. We now need to stall the processor for 1 cycle for all instructions.
B. The mistake can be corrected by adding an additional forwarding path from WB stage to ALU stage.
C. This error introduces only WAR hazard into the pipeline processor.
D. This error introduces both WAR and WAW hazards into the pipeline processor.
E. None of the above

6. Suppose there are two processor implementations:

   Processor A: Non-pipelined, single cycle implementation. Cycle time is 12ns.
   Processor B: 6 stages pipelined implementation. Cycle time is 3ns.

   Which of the following statement(s) regarding processor A and B is/are **TRUE**?

   i. Each instruction takes 18ns to execute on Processor B.
   ii. The latch delay time for Processor B must be 1ns per stage.
   iii. The speedup achieved by Processor B over Processor A is 12ns / (3ns*6).

   A. Only (i) is TRUE.
   B. Only (i) and (ii) are TRUE.
   C. Only (ii) and (iii) are TRUE.
   D. (i), (ii) and (iii) are all TRUE.
   E. None of the above

7. Which of the following statement(s) regarding "Delayed Branching" is/are **TRUE**?

   i. Delayed branching works in the same way as the "predict not taken" branch prediction.
   ii. With delayed branching, performance loss due to branch instructions can be removed completely.
   iii. Delayed branching changes the semantic (meaning) of the original program.

   A. Only (i) is TRUE.
   B. Only (i) and (ii) are TRUE.
   C. Only (ii) and (iii) are TRUE.
   D. (i), (ii) and (iii) are all TRUE.
   E. None of the above

8. Given the following jump (`j`) instruction in MIPS. Note that the processed immediate field is given instead of the jump target label.

   `j 0x20` # 0x20 is the 26-bit immediate, Instruction Address = **0xA0 00 00 40**

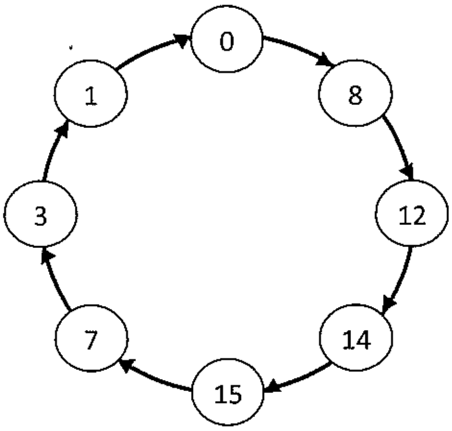
   Which of the following branch instructions can replace the `j` instruction and give the exact same behaviour? Note that the immediates are given in decimal for the answers.

   A. **`beq $0, $0, -33`**
   B. **`beq $0, $0, -17`**
   C. **`beq $0, $0, 15`**
   D. **`beq $0, $0, 60`**
   E. **`beq $0, $0, 64`**

9. **[10 marks]**

The following 4-bit Johnson counter *ABCD* is a sequential circuit that could be easily implemented using four *D* flip-flops as shown in class.

A	*B*	*C*	*D*
0	0	0	0
1	0	0	0
1	1	0	0
1	1	1	0
1	1	1	1
0	1	1	1
0	0	1	1
0	0	0	1



The state diagram is shown above with each state value written in decimal. We shall refer to the states with these decimal values for parts (b) to (d).

a. If the 8 states { 0000, 1000, 1100, 1110, 1111, 0111, 0011, 0001 } are codewords in a certain code, can this code detect 1-error? Explain your answer with suitable example. (No mark will be awarded for no or wrong explanation.) [1 mark]

b. Implement the above counter using *D* flip-flops for outputs *A* and *B*, a *T* flip-flop for output *C*, and a *JK* flip-flop for output *D*. Write out the **simplified SOP expressions** for all the flip-flop inputs. You do not need to draw the logic diagram. [5 marks]

c. What is the fewest number of logic gates required to implement the above counter? What are the logic gates required? [2 marks]

d. If the counter somehow lands in the unused state 2, what is the next state the counter will get into? [2 marks]

10. **[6 marks]**
You are to design a converter that takes in 4-bit input *ABCD* and generates a 3-bit output *FGH* as shown in Table 1 below.

Input				Output		
A	*B*	*C*	*D*	*F*	*G*	*H*
0	0	0	0	0	0	0
1	0	0	0	0	0	1
1	1	0	0	0	1	0
1	1	1	0	0	1	1
1	1	1	1	1	0	0
0	1	1	1	1	0	1
0	0	1	1	1	1	0
0	0	0	1	1	1	1

Table 1

S	$Y_3Y_2Y_1Y_0$
0	$J_3J_2J_1J_0$
1	$K_3K_2K_1K_0$

Table 2

You are given the following components:

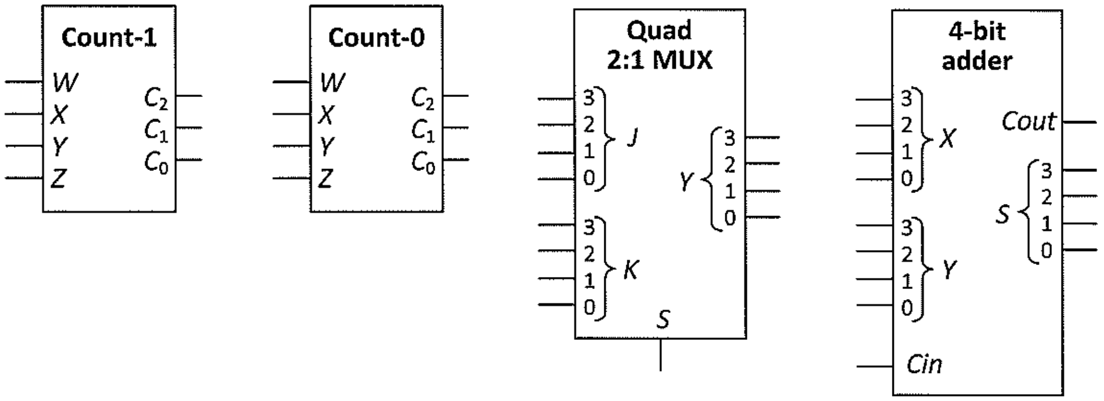
a. A **Count-1** device that takes in a 4-bit input *WXYZ* and generates a 3-bit output $C_2C_1C_0$ which is the number of 1s in the input. For example, if *WXYZ* = 0111, then $C_2C_1C_0$ = 011 (3).

b. A **Count-0** device that takes in a 4-bit input *WXYZ* and generates a 3-bit output $C_2C_1C_0$ which is the number of 0s in the input. For example, if *WXYZ* = 0111, then $C_2C_1C_0$ = 001 (1).

c. A **quad 2:1 multiplexer** that takes in two 4-bit inputs $J_3J_2J_1J_0$ and $K_3K_2K_1K_0$, and directs one of the inputs to its output $Y_3Y_2Y_1Y_0$ depending on its control signal *S*, as shown in Table 2 above.

d. A **4-bit parallel adder** that takes in two 4-bit unsigned binary numbers and outputs the sum.

The block diagrams of these components are shown below:



Given the above 4 components, you are to employ block-level design to design the converter, <u>without using any additional logic gate or other devices</u>. You observe that if *A* = 1, then the output *FGH* is simply the number of 1s in the input *ABCD*. You are to make your own observation for the case when *A* = 0.

11. **[8 marks]**
Given the following cycle time for the 5-stages MIPS processor:

	Fetch	**Decode/ Operand Read**	**ALU**	**Memory**	**Result Write**
Time Taken	**2 ns**	**1 ns**	**2 ns**	**3 ns**	**2 ns**

Suppose the following MIPS instructions are to be executed:

```
lw   $t0, 0($s0)
lw   $t1, 4($s0)
lw   $t2, 8($s0)
add  $t3, $t1, $t2
sub  $t0, $t0, $t3
sw   $t0, 12($s0)
```

For each of the following parts, give the **cycle time** (in ns), **total number of cycles** and **total execution time** (in ns). Timing chart is provided at the bottom of this page for your draft work.

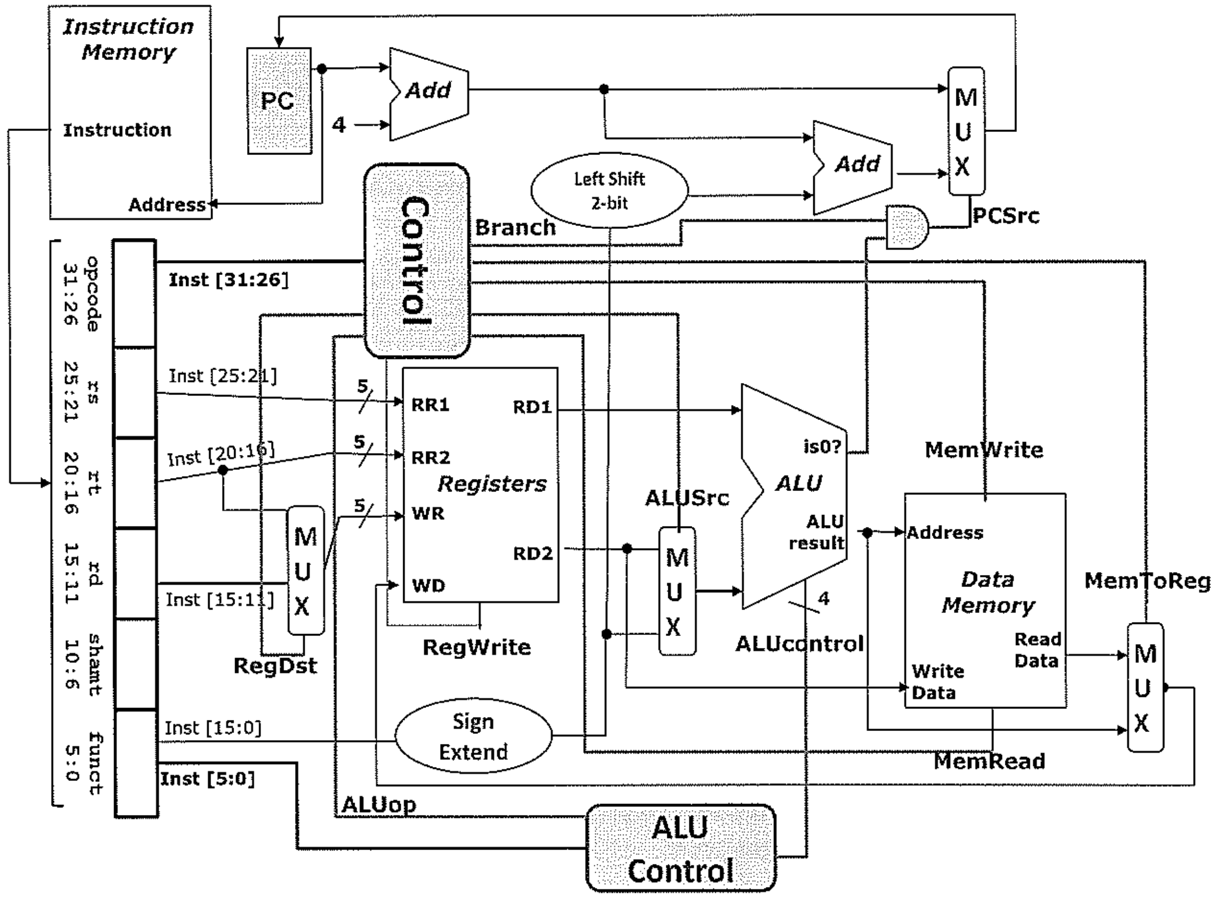
a. Non-pipelined, single cycle processor. [1 mark]

b. Pipelined processor with no forwarding. Register file supports write then read in a single cycle. [2 marks]

c. Pipelined processor with both ALU and Memory forwarding. Register file supports write then read in a single cycle. [2 marks]

d. **Reorder** the code to improve on the performance of (c). Give the reordered code and the improved execution time. If it is not possible to improve on the performance of (c), briefly explain why. [3 marks]

12. [**10 marks**]
Given the following datapath and control design for the MIPS processor, answer the following questions.



For your reference, the instruction formats used in this question are given below:

31	25	20	15	10	5	
opcode	rs	rt	rd	shamt	funct	R-type
opcode	rs	rt	immediate			I-type

a. If the following instruction is now under execution:

```
sb $5, 30($2)
```

What are the values of the control signals **RegDst**, **RegWrite**, **ALUsrc**, **MemRead**, **MemWrite** and **MemToReg**? Please note that the **MemToReg** multiplexer is the only one with swapped input (1 is the upper input, 0 the lower input). Use 'X' for don't care value. [3 marks]

b. If we want to support the "**xor**" MIPS instruction, which is an R-Format instruction with 0_{16} and 26_{16} as the Opcode and Function Code field. Indicate **only the signals which will be affected**, i.e. the combinatorial circuit that generates the control signal needs to be redesigned. There is no need to give the updated circuit design. [1 mark]

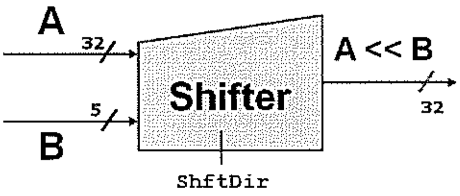
Q12. (continued)

c. Let us consider the task of incorporating the "**`sll`**" (shift-left logical) instruction into the MIPS processor. Suppose we have decided to include an additional datapath element ("Shifter"), which has the following functionality:

Inputs:
A is a 32-bit value
B is a 5-bit unsigned value

Output:
A << B, i.e. A left shifted by B positions

Control (ShftDir):
0 for left-shift, 1 for right-shift
Hardwired to 0 for left-shifting for this part



For the inputs A and B, indicate the correct values to be connected to them.

For the output (A<<B), indicate **an existing multiplexer** that it can connect to. Briefly explain how should we control that existing multiplexer properly. You do not need to indicate how the control signal(s) is/are to be generated. [4 marks]

d. It is obviously a waste to use the "shifter" element in (c) for left-shifting only. Suppose we want to incorporate the "**`srl`**" (shift-right logical) into the processor design in (c). Considering **only** the control signal (**`ShftDir`**), show how we can generate the signal from the Opcode/Funct Field of the "**`sll`**" and "**`srl`**" instructions. Please give a simplified Boolean expression only.

`sll` = 0_{16} / 0_{16} (opcode / funct field)
`srl` = 0_{16} / 2_{16} (opcode / funct field)

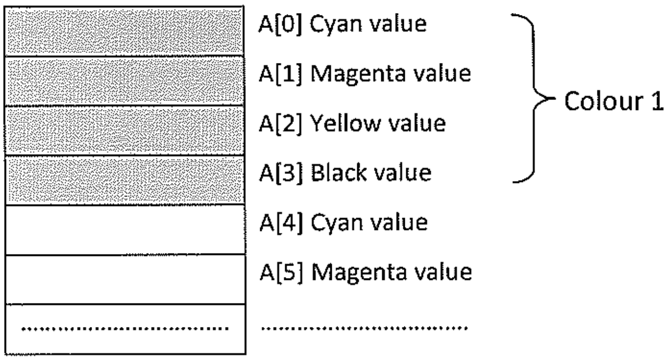
You can use Op[5:0] and F[5:0] to indicate the respective bits in the opcode and funct field. [1 mark]

e. How does the processor with parts (c) and (d) handle non-shift instructions? Briefly explain. [1 mark]

13. **[8 marks]**

(Background info) In computer program, colour can be represented using either RGB (Red, Green, Blue) or the CMYK (Cyan, Magenta, Yellow, Black) model. In CMYK model, a colour is represented by 4 values representing the saturation (how "deep" is the colour) of the four main colours: Cyan, Magenta, Yellow and Black.

Suppose we have stored the CMYK values of 16 colours as separate 32-bit integers in an array of size 64. So, the first four array elements (A[0...3]) represent the CMYK values for the first colour, the second set of four array elements (A[4....7]) represents the second colour, etc, as illustrated below:



Let's imagine that we need to manipulate the CMYK values separately for the 16 colours. There are 2 possible code fragments X and Y in some C-like high level programming language:

Code X:

```
//Each "int" is 32-bit
int A[64] = { ......... };

//Cyan values, A[0], A[4], ...
for (i = 0; i < 64; i = i + 4)
    Changes A[i]

//Magenta values, A[1], A[5], ...
for (i = 1; i < 64; i = i + 4)
    Changes A[i]

//Yellow values, A[2], A[6], ...
for (i = 2; i < 64; i = i + 4)
    Changes A[i]

//Black values, A[3], A[7], ...
for (i = 3; i < 64; i = i + 4)
    Changes A[i]
```

Code Y:

```
//Each "int" is 32-bit
int A[64] = { ......... };

//Go through 16 colours
for (i = 0; i < 16; i = i + 1) {

   for (j = 0; j < 4; j = j + 1) {
   //Go through the 4 values CMYK
        Changes A[i*4+j]
   }

}
```

We would like to find out whether the two different codes result in different memory access performance. For simplicity, the base of array A is assumed to be in memory location 0x0. You may also ignore the impact of variable "i" on cache access in the following questions.

Parts (a) and (b) refer to code X.

a. (Code X) Given a tiny **direct mapped cache** with 2 blocks of 8 bytes each. Give a tally of the following information: The number of cold/compulsory cache misses, and the number of conflict misses. [2 marks]

b. (Code X) If the cache hit time is 1ns and miss penalty is 10ns, what is the average cache access time in ns? [1 mark]

Parts (c) and (d) refer to code Y.

c. (Code Y) Given a tiny **direct mapped cache** with 2 blocks of 8 bytes. Give a tally of the following information: The number of cold/compulsory cache misses, and the number of conflict misses. [2 marks]

d. (Code Y) If the cache hit time is 1ns and miss penalty is 10ns, what is the average cache access time in ns? [1 mark]

Parts (e) and (f) refer to both codes X and Y.

e. If we increase the block size to a **direct mapped cache with 2 blocks** of 16 bytes each, which of the code fragments would see an improvement in average cache access speed (compared to (b) and (d))? You can indicate "none" if there is no improvement for both code fragments. [1 mark]

f. If we change to a **2-way set associative cache with 4 blocks of 8 bytes each**, which of the code fragments would see an improvement in average cache access speed? You can indicate "none" if there is no improvement for both code fragments. [1 mark]

**~~~ END OF PAPER ~~~**