# NATIONAL UNIVERSITY OF SINGAPORE

## SCHOOL OF COMPUTING

### EXAMINATION FOR
### Semester 2 AY2009/2010

## CS2100 – COMPUTER ORGANISATION

**April 2010** **Time allowed: 2 hours**

---

## INSTRUCTIONS TO CANDIDATES

1. This examination paper consists of **FIFTEEN (15)** questions and comprises **NINE (9)** printed pages.
2. This is an **OPEN BOOK** examination.
3. Answer all questions.
4. Write your answers in the **ANSWER SCRIPT** provided.
5. Fill in your Matriculation Number with a pen, clearly on every page of your ANSWER SCRIPT.
6. You may use pencil to write your answers.
7. You are to submit only the ANSWER SCRIPT and no other document.

## SECTION A (10 Multiple Choice Questions: 20 Marks)

Each question has one correct answer. Write your answer in the space provided in the ANSWER SCRIPT. Two marks are awarded for each correct answer and no penalty for wrong answer. You are advised not to spend more than 20 minutes on this section.

1. The 1st through 5th values in the standard 8-bit Gray code sequence are: 00000000, 00000001, 00000011, 00000010, and 00000110. What are the 6th value and the 200th value?

   A. 00000100 and 10101100
   B. 00000101 and 10100100
   C. 00000101 and 10101100
   D. 00000111 and 10100100
   E. 00000111 and 10101100

2. Which of the following MIPS instructions use the I-format?

   (i) `addi $s0, $s1, -100`
   (ii) `beq $t0, $t1, Loop`
   (iii) `srl $t2, $t3, 8`
   (iv) `lw $s1, -12($t2)`

   A. Only (i) and (ii)
   B. Only (i) and (iv)
   C. Only (i), (ii) and (iii)
   D. Only (i), (ii) and (iv)
   E. All of them use the I-format.

3. What architecture does the following code most likely resemble, where @*n* refers to a memory location associated with variable *n*. The code corresponds to the C statement: **e = (a + b) * c + d;**

```
load  @a
add   @b
mul   @c
add   @d
store @e
```

   A. Stack
   B. Accumulator
   C. Memory-memory
   D. Register-register
   E. All of the above.

4. An ISA has two types of fixed-length instructions: one has 4-bit opcode and the other has 6-bit opcode. Assuming that both types exist and the encoding space for opcode is completely utilized, what are the minimum total number of instructions and the maximum total number of instructions?

   A. Minimum = 2; Maximum = 80
   B. Minimum = 16; Maximum = 64
   C. Minimum = 19; Maximum = 61
   D. Minimum = 20; Maximum = 60
   E. None of the above.

5. Given two implementations M1 and M2 of the same instruction set, the table below shows the number of cycles for each instruction class in both machines and their frequencies for a typical program.

| ***Instruction class*** | ***CPI on M1*** | ***CPI on M2*** | ***Frequency*** |
|---|---|---|---|
| A | 1 | 2 | 50% |
| B | 3 | 3 | 30% |
| C | 5 | 4 | 20% |

   Assume that M1 has a clock rate of 80MHz and M2 has a clock rate of 100MHz. Which machine has a higher instructions-per-second rating and by how much?

   A. M1 is faster by 1.125
   B. M2 is faster by 1.125
   C. M1 is faster by 1.111
   D. M2 is faster by 1.111
   E. None of the above.

6. Suppose we can improve the floating point instruction performance of a machine by a factor of 12 (the same floating point instructions run 12 times faster on this new machine), what percent of the instructions must be floating point to achieve an overall speedup of 4?

   A. 80.36%
   B. 81.82%
   C. 83.33%
   D. 90.00%
   E. 92.50%

7. We may use the pseudo-instruction "`la`" in PCSPIM to load the address of some data, as shown in the code snippet below:

```
      .data
msg: .asciiz "Hello"
      .text
         . . .
      la $a0, msg
```

The "`la`" pseudo-instruction is converted into one or more real instructions. Which of the following real instructions appears in the converted code?

A. `addi`
B. `and`
C. `lw`
D. `li`
E. `lui`

8. If the rotate left pseudo-instruction "`rol rdest, rsrc1, rsrc2`" is made into a real instruction, what would be the values of control signals RegDst, RegWrite and MemtoReg?

A. RegDst = 1; RegWrite = 1; MemtoReg = 0
B. RegDst = 1; RegWrite = 1; MemtoReg = 1
C. RegDst = 1; RegWrite = 0; MemtoReg = 0
D. RegDst = 0; RegWrite = 1; MemtoReg = 0
E. RegDst = 1; RegWrite = 0; MemtoReg = 1

9. Assume a 5-stage pipeline. If each stage takes 3ns, and there are 20 instructions, what is the speedup for a pipelined system versus a non-pipelined single-cycle system?

A. 3.333
B. 4.000
C. 4.167
D. 4.667
E. 5.000

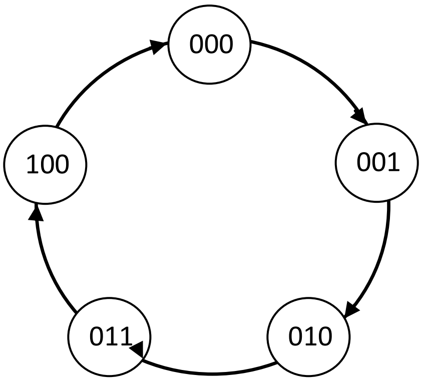
10. Which of the following claims that the number of transistors on an integrated circuit has doubled approximately every two years (later revised to every 18 months)?

    A. Moore's law
    B. Amdahl's law
    C. DeMorgan's law
    D. Turing's law
    E. Bill Gates' law

---

## SECTION B (5 Questions: 80 Marks)

11. [12 marks]
    The state diagram below shows a sequential circuit with states *ABC* cycling through the states 000, 001, 010, 011, 100 (and back to 000). *A* is implemented with a *D* flip-flop, *B* with a T flip-flop, and *C* with a JK flip-flop.



    a. Complete the state table in the answer script. [6 marks]
    b. Write the simplified SOP expressions for *DA*, *TB*, *JC* and *KC*. [3 marks]
    c. If the sequential circuit is designed using the simplified SOP expressions obtained in part (b) above, complete the state diagram by drawing an outflowing arrow from each of the 3 unused states. [3 marks]

12. [15 marks]
The **2-out-of-5 code** is a 5-bit code where each code value contains exactly two 1s. For example, 01010 is valid but 10110 and 00100 are not.

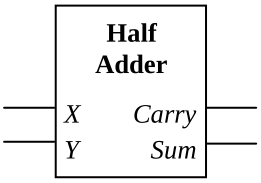
   a. List out all possible code values. [2 marks]

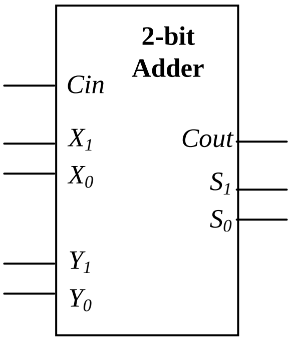
   b. Explain why the code is useful for transmission of numeric data. [2 marks]

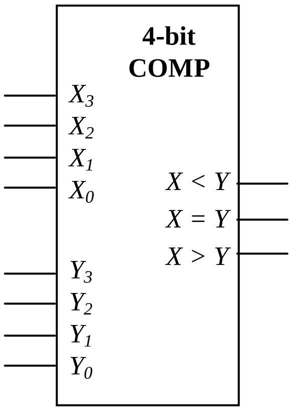
   c. What is the Hamming distance for this code? [2 marks]

   d. The code is represented by *abcde*, and the Boolean function *V(a,b,c,d,e)* is 1 if the code is valid, or 0 otherwise. Write the simplified SOP expression for *V*. [2 marks]

   e. Given a 4-bit magnitude comparator, a 2-bit parallel adder, and some half-adders, without any other device or logic gate, implement the function *V* as described in part (d) above. The block diagrams of the devices are given below. [7 marks]







13. [20 marks]
An algorithm to find the *greatest-common-divisor* (GCD) of two positive integers *a* and *b* is shown below:

```
while ($b \neq 0$) {
        $r = a \% b$;  // $r$ is the remainder of $a/b$
        $a = b$;
        $b = r$;
}
Report $a$, the answer
```

The **div**, **mfhi** and **mflo** instructions are explained below (you may not need to use all of them). You are to use this **div** instruction, not the **div** pseudo-instruction with 3 operands (i.e., div rdest, rsrc1, rsrc2).

`div rs, rt`

| 0 | rs | rt | 0 | 0x1a |
|---|---|---|---|---|
| 6 | 5 | 5 | 10 | 6 |

Divide register **`rs`** by register **`rt`**. Leave the quotient in register **`lo`** and the remainder in register **`hi`**.

`mfhi rd`

| 0 | 0 | rd | 0 | 0x10 |
|---|---|---|---|---|
| 6 | 10 | 5 | 5 | 6 |

Copies the value in the **`hi`** register to register **rd**.

`mflo rd`

| 0 | 0 | rd | 0 | 0x12 |
|---|---|---|---|---|
| 6 | 10 | 5 | 5 | 6 |

Copies the value in the **`lo`** register to register **rd**.

a. Translate the given GCD algorithm into MIPS code. The first few lines of the code are given in the answer script.

You are to use **$s0** for *a*, and **$s1** for *b*. You are to print the answer on the console window. You should have a branch instruction and a jump instruction in order to answer parts (c) and (d) below. You should not use any pseudo-instruction except **li**. [12 marks]

b. Write the hexadecimal representation of the **div** instruction you used in your MIPS code in part (a). [3 marks]

c. What is the immediate value (in hexadecimal) in the branch instruction in your MIPS code? [2 marks]

d. If the first instruction (li $v0, 5) is at address **00400024** (in hexadecimal), what is the immediate value (in hexadecimal) in the jump instruction? [3 marks]

14. [16 marks]

The following MIPS code generates numbers in the Fibonacci sequence (1, 1, 2, 3, 5, 8, 13, ...) and stores them into an integer array *A*, where the base address of *A* is in **$s0**. Each array element occupies 4 bytes.

```
      addi $t0, $zero, 1     # Instr1
      addi $s7, $s0, 80      # Instr2

      sw   $t0, 0($s0)       # Instr3
      add  $t1, $t0, $zero   # Instr4
      sw   $t1, 4($s0)       # Instr5
      addi $s1, $s0, 8       # Instr6
Loop: add  $t2, $t1, $t0     # Instr7
      sw   $t2, 0($s1)       # Instr8
      add  $t0, $t1, $zero   # Instr9
      add  $t1, $t2, $zero   # Instr10
      add  $s1, $s1, 4       # Instr11
      bne  $s1, $s7, Loop    # Instr12
```

a. How many elements of array *A* are written? [2 marks]

For parts (b) to (d) below, refer to the third sw instruction: `sw $t2, 0($s1)`

b. What is the register number supplied to the register file's "**Read register 1**" input? Is this register actually read? [2 marks]

c. What is the register number supplied to the register file's "**Read register 2**" input? Is this register actually read? [2 marks]

d. What is the register number supplied to the register file's "**Write register**" input? Is this register actually written? [2 marks]

e. Assuming a 5-stage MIPS pipeline without forwarding, list out all the data dependencies in the code that will cause delay in the pipeline, and the delay amount for each of the dependencies. You may refer to the instructions by their numbers (Instr1 to Instr12) in your answer. [4 marks]

f. Assuming a 5-stage MIPS pipeline with forwarding, but no branch prediction (however, branch is resolved in the ID stage), list out all the data and control dependencies in the code that will cause delay in the pipeline, and the delay amount for each of the dependencies. You may refer to the instructions by their numbers (Instr1 to Instr12) in your answer. [4 marks]

15. [17 marks]

A machine with 4K bytes of memory has a word size of 4 bytes, and a two-way set associative cache of 4 sets. Each block in the cache contains 2 words.

a. Write out the width (number of bits) of the tag, index, and offset fields. [3 marks]

b. You are given a sequence of 10 memory references as shown in the table below, where each memory reference is a byte address in hexadecimal format.

| | Memory address (in hexadecimal) |
|---|---|
| 1. | **03A** |
| 2. | **980** |
| 3. | **8C7** |
| 4. | **2F4** |
| 5. | **E5E** |
| 6. | **038** |
| 7. | **9C6** |
| 8. | **2F5** |
| 9. | **8C1** |
| 10. | **11B** |

Fill in the contents of the cache, assuming that it is initially empty. Use the notation **M[*a*]** to denote the word starting at memory address *a*, where *a* is in hexadecimal. Use hexadecimal notation for the values in the tags as well.

If both blocks in a set are empty, fill in the left block first. If both blocks in a set are filled, replace the one that is least recently accessed, that is, apply the **LRU replacement policy**. When you replace a block, cross out the content and fill in the new content below the old one. [10 marks]

c. How many cache hits are there? [1 mark]

d. (This is a general question and is not related particularly to the above parts.) For the following three types of cache miss, which might cause the least delay and which might cause the most delay in general? Explain your answers.

- Cache read miss from a data cache
- Cache write miss from a data cache
- Cache read miss from an instruction cache

[3 marks]

**=== END OF PAPER ===**