NATIONAL UNIVERSITY OF SINGAPORE

With answers.

SCHOOL OF COMPUTING

TERM TEST #2
AY2012/3 Semester 2

CS2100 — COMPUTER ORGANISATION

13 April 2013 Time Allowed: **1 hour 15 minutes**

INSTRUCTIONS

1. This question paper contains **TEN (10)** questions and comprises **TEN(10)** printed pages.
2. The last three pages are for your reference and rough work. They contain the "datapath and control" diagram and empty timing charts.
3. An **Answer Sheet**, comprising **TWO (2)** printed page, is provided for you.
4. Answer **ALL** questions within the space provided on the **Answer Sheet**.
5. Maximum score is **30 marks**.
6. This is a **CLOSED BOOK** test. However, a handwritten single-sheet double-sided A4 cheat sheet is allowed.
7. Write legibly with a pen or pencil (at least 2B).
8. Calculators and computing devices such as laptops and PDAs are not allowed.
9. Submit only the **Answer Sheet** at the end of the test. You may keep the question paper.
10. Write your **MATRICULATION NUMBER** on the **Answer Sheet** using a **PEN**.

——— END OF INSTRUCTIONS ———

SECTION A (6 Multiple Choice Questions: 12 marks)

Each question has only one correct answer. Write your answer in the space provided on the **Answer Sheet**. Two marks are awarded for each correct answer and no penalty for wrong answer.

1. Suppose there are 20 memory load instructions in a program with 100 instructions and we reduce the cycles required for the load instruction by 50%. What is the speedup for this program?

   A. 1.2 (20% improvement)
   B. 1.5 (50% improvement)
   C. 1.1 (10% improvement)
   D. Not enough information for a definite answer
   E. None of the above

   20 instructions != occupies 20% of execution time.
   As the number of cycles is not known, the effect of the speedup cannot be calculated properly.

2. Given that the opcode for "addi" is $\mathbf{8_{16}}$. The instruction encoding (in hexadecimal) for "**`addi $2, $4, -3`**" is:

   A. **`0x20 82 00 03`**
   B. `0x20 82 FF FD`
   C. **`0x20 44 00 03`**
   D. **`0x20 44 FF FD`**
   E. None of the above

   001000 00100 00010 1111111111111101
   = 0010 0000 1000 0010 1111 1111 1111 1101
   = 2 0 8 2 F F F D

3. The pseudo-instruction "**`ble $s0, $s1, there`**" (branch-less-or-equal) can be translated to:

   A.
   ```
   slt $t0, $s0, $s1
   beq $t0, $zero, there
   ```

   B.
   ```
   slt $t0, $s0, $s1
   bne $t0, $zero, there
   ```

   C.
   ```
   slt $t0, $s1, $s0
   beq $t0, $zero, there
   ```

   D.
   ```
   slt $t0, $s1, $s0
   bne $t0, $zero, there
   ```

   E. None of the above

   less-or-equal == ! (greater than)

4. Given a **4-stage single cycle processor** with the following information:

	Stage 1	Stage 2	Stage 3	Stage 4
Time Needed	**2ns**	**1ns**	**3ns**	**2ns**

How much time is needed to execute 100 instructions on this processor?

A. 100ns
B. 200ns
C. 800ns
D. 1200ns
E. None of the above

Cycle time = sum of all stages = (2+1+3+2) = 8ns
Total Cycles = #inst * 1 cpi = 100
Time = 100 * 8 = 800ns

5. This question uses the same information given in Q4. Suppose we change the processor to a pipeline design. Three pipeline latches are added, each with a delay of 1ns. What is the total time needed to execute 100 instructions on this new processor? You can assume the instructions are free of instruction dependencies (i.e. no hazards).

A. 309ns
B. 400ns
C. 412ns
D. 416ns
E. None of the above

Pipeline cycle time = 3 (max of Ti) + 1 (Td) = 4ns

Total Cycles = (#inst + N-1) = 100 + 3 = 103
Time = 103 * 4 = 412ns

6. Given the MIPS instruction "**`beq $7, $0, -10`**" at address 0x400040, where can we find the branch target in memory?

A. **`0x40 00 18`**
B. `0x40 00 1C`
C. **`0x40 00 30`**
D. **`0x40 00 31`**
E. None of the above

offset = (PC+4) + (-10*4 bytes) = -9 instructions = -36 bytes = -24_{16}

address = 0x40001C

SECTION B (Bonus Question: 1 mark)

This is a bonus question. The mark of this question will be added only if the total mark scored is less than 30.

7. The lecturer loves to use strange analogies in the lecture, which of the following **is NOT used?**

   A. Uncle Soo rolls down the slopes while jogging.
   B. Lord of the rings.
   C. Falling in love with a computer.
   D. Ostrich sticking its head in the sand.
   E. Harry Potter.

A. Amdahl's Law
B. Datapath and Control
C. End of MIPS
D. Pipeline Hazards

SECTION C (Questions: 18 marks)

Write your answer in the space provided on the **Answer Sheet**. You do not need to show workings, unless otherwise stated.

8. Take a look at the following MIPS program and the memory content. **For simplicity, all values (memory address and content) are in decimal. [4 marks]**

```
#s1 is initialized to 0
#t0 is initialized to 112

loop:
      beq $t0, $zero, exit
      lw  $t1, 0($t0)
      add $s1, $s1, $t1
      lw  $t0, 4($t0)
      j   loop
exit:
```

Address	Content
100	120
104	132
108	128
112	108
116	124
120	116
124	104
128	100
132	136
136	112

Suppose we execute the code for **3 iterations**, answer the following:

(a) Give the value of register $s1 at the end of the 3rd iteration. **[2 marks]**
Ans: 108 + 104 + 120 = 332

(b) If we want to terminate the loop at the 4th check of "beq", some memory contents need to be modified. Give the **address(es)** and the **modified content(s)** to achieve this. You should find the minimum changes required. **[2 marks]**
Ans: Change the content of location 104 to 0.

Background info: The code is the frequently used "linked list hopping" code

```
while ( ptr != NULL ){
    sum += ptr->item;
   ptr = ptr->next;
}
```

Marking Note:
Content = 0 (1 mark)
Address + Content = 2 (marks)
Any additional answers = -1 mark

9. Suppose the following code is executed on a 5-stage MIPS pipeline processor. **[5 marks]**

`add $12, $15, $23`	`#i1`
`lw  $10, 24($12)`	`#i2`
`sub $7, $12, $10`	`#i3`
`add $9, $10, $23`	`#i4`
`or  $9, $3, $5`	`#i5`
`sw  $9, 24($12)`	`#i6`

(Note: Use the timing chart on page 10 to help.)

(a) Let us label the two forwarding paths as follows:

A. From EX/MEM pipeline latch to ALU unit

B. From MEM/WB pipeline latch to ALU unit

For each of the instructions, indicate which receiving instruction(s) it passes the result to, and via which forwarding path(s). For example:

	Forward to	Via
#i7	#i9	A

Indicates instruction #i7 uses forwarding path "(**A**) From EX/MEM to ALU unit" to pass result to instruction #i9. **Note that you need to get both parts correct to earn marks. [4 marks]**

Ans:

	Forward to	**Via**	**Marking Notes**
#i1	**#i2**	**A**	**1 mark**
#i2	**#i3**	**B**	**2 marks**
#i3			
#i4			
#i5	**#i6**	**A**	**1 mark**
#i6			

Marking Note: ANY additional RAW written will get -1 mark (one time penalty, not for each pair).

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
i1	**F**	**D**	**E**	**M**	**W**														
i2		**F**	**D**	**E**	**M**	**W**						CC4: #i1 forward $12 to #i2							
i3			**F**	**F**	**D**	**E**	**M**	**W**				CC6: #i2 forward $10 to #i3							
i4					**F**	**D**	**E**	**M**	**W**			RAW with #i2 handled by Reg.File's "write-then-read"							
i5						**F**	**D**	**E**	**M**	**W**									
i6							**F**	**D**	**E**	**M**	**W**	CC9: #i5 forward $9 to #i6							

(b) How many cycles are needed for the code if all forwarding paths in (a) are implemented? **[1 mark]**

Ans: 11 cycles, See the given timing chart.

The I-format instructions in MIPS can expand the immediate field (16-bit) to a 32-bit value in two ways:

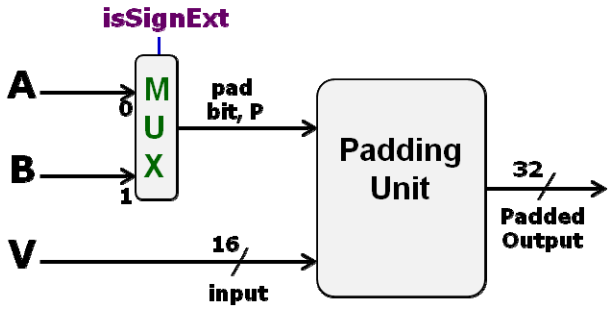
Expansion Type	Example Instructions	Expansion of the immediate field
Sign Extended	`addi $3, $7, 5` `addi $3, $7, -5`	0x00 05 ➔ 0x00 00 00 05 0xFF FB ➔ 0xFF FF FF FB
Zero Padded	`ori $3, $7, 0xFFFB`	0xFF FB ➔ 0x00 00 FF FB

Essentially, the logical operations (andi, ori, xori) covered in this course expand the immediate by performing zero-padding instead of sign extension.

Suppose we want to extend the simple MIPS datapath and control design to handle the set of logical operations (andi, ori and xori). This question will deal with only the modification required to handle the two types of immediate field expansion.

Let us upgrade the original datapath element that handles the expansion from:

to



The "padding unit" takes in a 16-bit input value **V** and a 1-bit pad bit **P**, it will then output a 32-bit value by padding the most significant 16 bits with the pad bit **P** and the lower 16 bit with the input value **V**. A multiplexer is added to handle both types of expansion with the same design. The multiplexer uses the control signal "isSignExt" to either select the upper port (0) or the lower port (1). The control signal should be a "1" when sign extension is required, "0" when zero padding is needed. [9 marks]

(a) Give the 1-bit value for the upper and lower ports of the multiplexor. You can use any value readily available in the datapath. No additional logic gate or datapath element is allowed. **[2 marks]**

Ans: upper port = 0, lower port = Inst[15] (1 mark each)

(b) Indicate an existing component that should handle the generation of the "isSignExt" signal. **[1 mark]**

Ans: Control Unit (The decision can be made purely by Opcode. Since Control unit is already handling all signals related to Opcode, it is the best choice.)

(c) Suppose the datapath elements have the following delay:

Inst-Mem / Data Mem	Adder / ALU Ctrl	MUX	ALU	Reg.File	Control	Left-shift / AND / Sign Ext / Padding Unit
200ps	30ps	10ps	60ps	100ps	70ps	20ps

(Please note the delay on the ALU Control Unit and the Padding Unit)

Which instruction(s) out of the list {**addi**, **beq**, **sw**} will require a **longer execution time compared to the original design? What is the additional duration required? [6 marks]**

(The complete datapath and control diagram is provided for your reference on page 9.)

Ans:

addi (blue font represents additional delays in the new design)

→ Control → M(isSignExt) → Pad → M(ALUScr)
→ ALU Ctrl
IM → Reg.File → ALU → M(MtoReg) → Reg.File

Original: 200 (IM) + 100 (Reg.File) + 60(ALU) + 10(M(MtoReg)) + 100(Reg.File)

New: 200 (IM) + 70 (Control) + 10(M(isSignExt) + 20(pad) + 10(M(ALUSrc) + 60(ALU) + 10(M(MtoReg)) + 100(Reg.File)

Difference = 10ps

---

beq (blue font represents additional delays in the new design)

→ Control → M(isSignExt) → Pad → Shift → Adder
→ ALU Ctrl
IM → Reg.File → M(ALUScr) → ALU → AND → M(PCSrc)

Original: 200 (IM) + 100 (Reg.File) + 10(M(ALUSrc)) +60(ALU) + 20(AND) + 10(M(PCSrc))

New: **Same.** The upper path is still much lesser than the lower path

---

sw (blue font represents additional delays in the new design)

→ Control → M(isSignExt) → Pad → M(ALUScr)
→ ALU Ctrl
IM → Reg.File → ALU → DM

Original: 200 (IM) + 100 (Reg.File) + 60(ALU) + 200(DM)

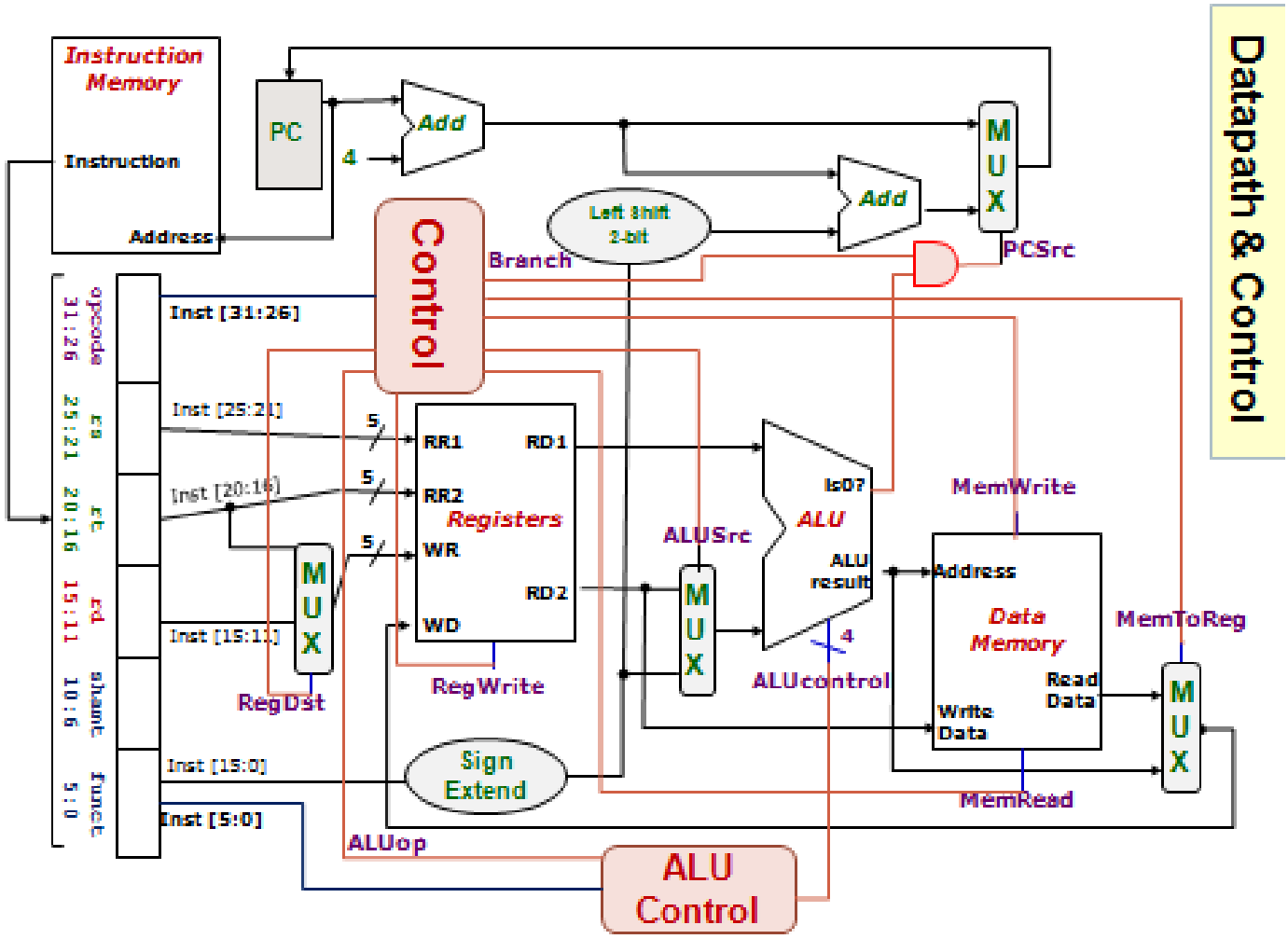
New: 200 (IM) + 70 (Control) + 10(M(isSignExt) + 20(pad) + 10(M(ALUSrc) + 60(ALU) + 200(DM)

Difference = 10ps

Marking Note: "longer time" = 1 mark, correct difference = 1 mark; "same" = 2 marks (for beq only).

——— END OF PAPER ———

This page is for your rough work.



Inst-Mem / Data Mem	Adder / ALU Ctrl	MUX	ALU	Reg.File	Control	Left-shift / AND / Sign Ext / Padding Unit
200ps	30ps	10ps	60ps	100ps	70ps	20ps