IT5002-Midterm-Ans - Midterm answer

Computer Systems and Applications (National University of Singapore)

IT5002 2021/22 Semester I
Backup Midterm Paper

1. This paper is only to be used in the event of a serious LumiNUS failure. Otherwise this paper should be completed online in the LumiNUS Quiz Tool.

2. This is an open-internet paper. You can use any tool (compilers, code converters, online forums, etc.), as long as you **DO NOT COMMUNICATE WITH ANYONE NOR RECEIVE COMMUNICATIONS FROM ANYONE. IF YOU DO SO YOU WILL BE DEEMED TO HAVE CHEATED AND SERIOUS DISCIPLINARY ACTION WILL BE TAKEN AGAINST YOU.**

3. You may ask questions at this link: https://forms.gle/bXuZiv5kkCtX84pv9. Your questions must be phrased in a way that can be answered with a "YES" or a "NO", with no further elaboration. If your question is not appropriately phrased, you may be asked to rephrase, or you may be ignored.

4. This paper consists of FOURTEEN (14) questions on FIVE (5) printed pages including this page. Marks are indicated, and total 40 marks, forming 20% of your final course grade.

5. You may complete this paper electronically on your tablet, print it out and write down your answers, or simply write your answers on a blank piece of paper. If you use a blank piece of paper, ensure that your answers are properly labeled so that we can identify which question (or which part of a question) your answer belongs to.

6. This paper is 60 minutes long.

7. **ENSURE THAT YOUR NAME, STUDENT NUMBER AND TUTORIAL GROUP NUMBER ARE CLEARLY WRITTEN ON YOUR ANSWER SHEET. IF YOU FAIL TO WRITE ANY OF THESE YOUR PAPER MAY NOT BE GRADED AND YOU WILL RECEIVE ZERO FOR THIS ASSESSMENT!**

8. At the end of the assessment, submit your answers as a PDF named TYY_Axxxxxx.pdf, where TYY is your tutorial group number, and Axxxxxx is your student number. **IF YOU SUBMIT USING AN INCORRECT NAMING, YOUR SCRIPT MAY NOT BE GRADED AND YOU WILL RECEIVE ZERO FOR THIS ASSESSMENT!**

9. IF LUMINUS IS AVAILABLE, Submit to your personal folder in the MIDTERMS->Txx folder, where Txx is your tutorial group number. If LumiNUS is not available, email your script, PROPERLY NAMED TYY-Axxxxxx.pdf, to cs2100.papers@gmail.com. Here TYY is your tutorial group number and Axxxxx is your student number. **REMINDER: IF YOU SUBMIT USING AN INCORRECT NAMING, YOUR SCRIPT MAY NOT BE GRADED AND YOU WILL RECEIVE ZERO FOR THIS ASSESSMENT!**

All rules in the Standard Operating Procedures are to be observed. Failure to do so may result in disciplinary action being taken against you.

NAME: _____ STUDENT ID: _____

TUTORIAL GROUP NO: _____

Questions.

Q1. What is the result of this subtraction in base 5? (2 marks)

$212_5 - 113_5 = \mathbf{44}_5$

Check:
$212_5 = 57$
113 = 33
$57 - 33 = 24 = 44_5$

Q2. We are given the following sum in an unknown base v. (2 marks)

$$\begin{array}{r} 213_v \\ +\ 124_v \\ \hline 340_v \end{array}$$

The unknown base v is base **7**

Check:
$213_7 = 108$
$124_7 = 67$
$108 + 67 = 175 = 340_7$

Q3. We are given the following subtraction in an unknown base w: (2 marks)

$$\begin{array}{r} 211_w \\ -\ 112_w \\ \hline 33_w \end{array}$$

The unknown base w is base **4**

Check:
$211_4 = 37$
$124_4 = 22$
$47\text{-}22 = 15 = 33_4$

Q4. We have the following conversion in an unknown base x: (1 mark)

$3233_x = 11101111_2$

The unknown base x is base ***4***

Check: $11101111_2 = 11\ 10\ 11\ 11_2 = 3\ 2\ 3\ 3_4$

For Q5 and Q6 we consider a 12-bit floating point number system, with one sign bit, 6 exponent bits in 2's complement, and 5 mantissa bits stored without any kind of rounding, with one hidden bit.

Q5. In this number system, the smallest (most negative) exponent possible is ***-32*** and the largest exponent possible is ***31***. (4 marks)

Check:

2^6 = 64 numbers, split into negative and non-negative ranges, so 32 numbers in each range. Negative range = -32 to -1, positive range = 0 to 31. Hence most negative is -32 and most positive is 31.

Q6. The hexadecimal representation of -23.203125 in this number system is 0x***88e***. Use as few hexadecimal digits as possible. (1 mark)

Check:

$-23.203125 = -10111.001101_2$
$= -1.0111001101_2 \times 2^4$
Mantissa = 01110
Exponent = 000100
Sign = 1
Binary: 1000 1000 1110 = 0x88E

Q7. We consider a 16-bit signed number system in excess-4096. In this a number system, the most negative number is ***-4096*** and the most positive number is ***12,287*** (1 mark)
Check:

In excess-4096, x + 4096 = 0, so x = -4096. Negative range is -4096 to -1 giving us 4096 numbers. There are 65536 numbers in a 16 bit binary sequence, so # of non-negative numbers = 65536 – 4096 = 61,440. This starts from 0, so the non-negative range goes from 0 to 64,439.

Q8. In this question we consider the 3-digit radix complement of a number in base 7, called the 3-digit 7s complement representation. (2 marks each = 4 marks)

Find the following two decimal numbers in 3-digit 7s complement:

112_{10} = ***220***$_{7s}$
-213_{10} = ***244*** $_{7s}$

Check:

220_{7s} = 112
244_{7s} = $-(423)_7$ = -213

Q9. Using your answers from the previous question, find the following in 3-digit 7's complement: (2 marks)

112 - 213 = ***464***$_{7s}$

Check:

$220_{7s} + 244_{7s} = 464_{7s} = -(203)_{7s} = -101_{10} = 112 - 213$

In Questions 10 to 13 we will consider the following MIPS assembly program. This program processes an array A of integers.

```
        addi $t0, $zero, 0
        addi $t1, $zero, 0
        addi $t2, $s3, 0
        addi $t3, $s4, -1
        sll $t3, $t3, 2
        add $t3, $s3, $t3
a:      slt $t4, $t1, $s4
        beq $t4, $zero, e
b:      slt $t4, $t2, $t3
        beq $t4, $zero, d
        lw $t4, 0($t2)
        lw $t5, 4($t2)
        slt $t6, $t4, $t5
        beq $t6, $zero, c
        sw $t4, 4($t2)
        sw $t5, 0($t2)
        addi $t0, $zero, 1
c:      addi $t2, $t2, 4
        j b
d:      beq $t0, $zero, e
        addi $t0, $zero, 0
        addi $t2, $s3, 0
        addi $t1, $t1, 1
        j a
e:
```

Q10. Choose ALL of the statements that are TRUE about this program. (2 marks)

a. **Array A's base address is in register $s3.**
b. Register $t1 contains the address of the current element from array A being processed.
c. Register $t3 will contain the address of the last element of the array.
d. **This program reads A[i] and A[i+1], and swaps them if A[i] < A[i+1]**
e. If Register $t0 is 1, the program exits.

When this program is run on an array with 5 elements, what is the MINIMUM number of instructions that will be executed? Answer: ***43*** instructions.

For the shortest execution count, there are no swaps at all, and the instructions that are cancelled below will not be executed.

```
        addi $t0, $zero, 0
        addi $t1, $zero, 0
        addi $t2, $s3, 0
        addi $t3, $s4, -1
        sll $t3, $t3, 2
        add $t3, $s3, $t3
a:      slt $t4, $t1, $s4
        beq $t4, $zero, e
b:      slt $t4, $t2, $t3
        beq $t4, $zero, d
        lw $t4, 0($t2)
        lw $t5, 4($t2)
        slt $t6, $t4, $t5
        beq $t6, $zero, c
        sw $t4, 4($t2)
        sw $t5, 0($t2)
        addi $t0, $zero, 1
c:      addi $t2, $t2, 4
        j b
d:      beq $t0, $zero, e
        addi $t0, $zero, 0
        addi $t2, $s3, 0
        addi $t1, $t1, 1
        j a
e:
```

The inner loop always executes one less times than the number of elements. Hence it will iterate 4 times for 5 elements. There are 8 instructions executed per iteration. On the 5th iteration, only the slt/beq at b: will be executed. Total instructions executed = 8 x 4 + 2 = 34 instructions.

```
b:      slt $t4, $t2, $t3
        beq $t4, $zero, d
        lw $t4, 0($t2)
        lw $t5, 4($t2)
        slt $t6, $t4, $t5
        beq $t6, $zero, c
        sw $t4, 4($t2)
        sw $t5, 0($t2)
        addi $t0, $zero, 1
c:      addi $t2, $t2, 4
        j b
```

On the outer loop:

```
a:  slt $t4, $t1, $s4
        beq $t4, $zero, e
```

(inner loop)

```
d:  beq $t0, $zero, e
        addi $t0, $zero, 0
        addi $t2, $s3, 0
        addi $t1, $t1, 1
        j a
e:
```

Notice that because $t0 is 0, we only execute the beq at d. The 4 instructions after that are not executed (and hence shown canceled here).

For the two loops, total number of instructions executed = 2 + (inner loop) + 1 = 2 + 34 + 1 = 37, and we add in the 6 instructions before a, giving us 43 instructions.

When this program is run on an array with 5 elements, what is the MAXIMUM number of instructions that will be executed? Answer: ***273*** instructions.

```
        addi $t0, $zero, 0
        addi $t1, $zero, 0
        addi $t2, $s3, 0
        addi $t3, $s4, -1
        sll $t3, $t3, 2
        add $t3, $s3, $t3
a:      slt $t4, $t1, $s4
        beq $t4, $zero, e
b:      slt $t4, $t2, $t3
        beq $t4, $zero, d
        lw $t4, 0($t2)
        lw $t5, 4($t2)
        slt $t6, $t4, $t5
        beq $t6, $zero, c
        sw $t4, 4($t2)
        sw $t5, 0($t2)
        addi $t0, $zero, 1
c:      addi $t2, $t2, 4
        j b
d:      beq $t0, $zero, e
        addi $t0, $zero, 0
        addi $t2, $s3, 0
        addi $t1, $t1, 1
        j a
e:
```

In the worst case, the swap and setting $t0 is always executed.

```
b:      slt $t4, $t2, $t3
        beq $t4, $zero, d
        lw $t4, 0($t2)
        lw $t5, 4($t2)
        slt $t6, $t4, $t5
        beq $t6, $zero, c
        sw $t4, 4($t2)
        sw $t5, 0($t2)
        addi $t0, $zero, 1
c:      addi $t2, $t2, 4
        j b
```

Thus the inner loop will execute 11 instructions for 4 iterations, giving 44 instructions. On the 5th iteration only the slt and beq at b: are executed, giving us a total of 44 + 2 = 46 instructions.

On the outer loop:

```
a:  slt $t4, $t1, $s4
        beq $t4, $zero, e
```

(Inner loop)

```
d:  beq $t0, $zero, e
        addi $t0, $zero, 0
        addi $t2, $s3, 0
        addi $t1, $t1, 1
        j a
e:
```

Here $t0 is never 0, hence all 5 instructions after the inner loop will be executed. The outer loop executes 5 times, and each time it will execute (2 + *inner loop* + 5) instructions = 2 + 46 + 5 = 53 instructions.

This iterates 5 times = 265 instructions. On the 6th iteration, only slt and beq at a: are executed, giving 265 + 2 = 267 instructions. Add in the 6 instructions before a gives us 267 + 6 = 273 instructions.

Q13. Suppose Array A contains the following elements (element 0 is on the left):

6, 5, 9, 2, 3

Write down the contents of Array A after running this program. (5 marks)

Element 0: ***9***

Element 1: ***6***

Element 2: ***5***

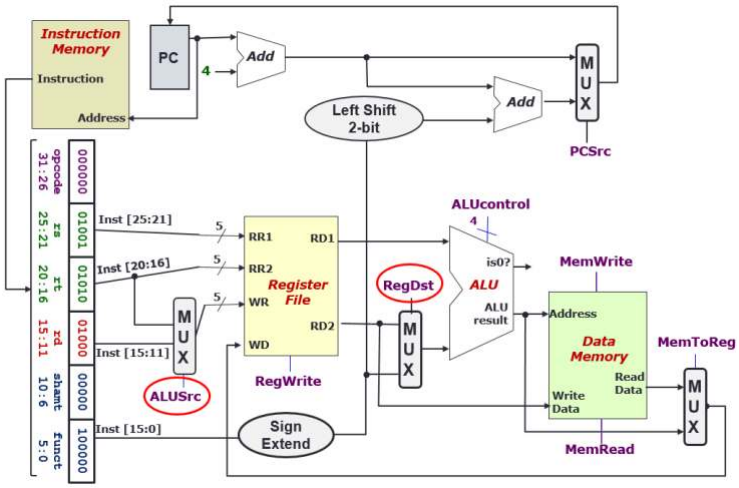
Element 3: ***3***

Element 4: ***2***

Q14.

In a previous tutorial, Mr. De Blunder had accidentally swapped the inputs of the RegDst multiplexer, resulting in incorrect execution.

That problem has now been fixed, but in the process Mr. De Blunder accidentally swapped the RegDst and ALUSrc signals; he wired the ALUSrc signal to the RegDst multiplexer, and the RegDst signal to the ALUSrc multiplexer! This blunder is shown below, with the swapped signals circled.



Assume that each register has been initialized to its own register number. That is, register $1 contains 1, $2 contains 2, $3 contains 3, etc. Note that for non-shift instructions, shamt (shift amount) is **always zero.**

What is the result written back at the end of each of the following operations? Write your answers in decimal. (2 marks each = 8 marks)

General strategy: ALUSrc now controls whether we write to rt or rd, while RegDst now controls whether the second input to the ALU is RD2 or sign extended immediate.

sub $2, $3, $2 $2 will contain ***-8223*** after execution.

Check:
sub $2, $3, $2
opcode = 000000
rs = $3 = 00011
rt = $2 = 00010
rd = $2 = 00010
shamt = 00000
func = 100010
Total instruction = 000000 00011 00010 00010 00000 100010 (blue part shows the last 16 bits)
This is an R-Type instruction, thus RegDst = 1 and ALUSrc = 0. Since ALUSrc now controls whether to write to rd or rt, here it will write to rt. Likewise RegDst is 1 and will pick the 16 bit constant (sign extended) instead of RD2.

The 16-bit constant is shown in blue above and evaluates to 4096 + 32 + 2 = 4130. Thus this instruction will take (3 – 4130) and write this to $2, giving $2 = -4127.

addi $5, $6, 7 $5 will contain ***5*** after execution.

Check:
addi $5, $6, 7
opcode = 001000
rs = $3 = 00110
rt = $2 = 00101
Immed = 0000 0000 0000 0111
Total instruction = 001000 00110 00101 0000 0000 0000 0111 (blue part shows the last 16 bits)
For addi, RegDst = 0 and ALUSrc = 1. Since ALUSrc now controls whether to write to rd or rt, here it will write to "rd", actually the first 5 bits of Immediate, which is 00000. Thus the instruction writes to $0 and $5 remains unchanged.

ori $7, $8, 14336 $7 will contain ***15*** after execution.

Check:
ori $7, $8, 14336
opcode = 001101
rs = $8 = 01000
rt = $7 = 00111
Immed = 0011 1000 0000 0000
Total instruction = 001101 01000 00111 0011 1000 0000 0000 (blue part shows the last 16 bits)
For ori, RegDst = 0 and ALUSrc = 1. Since ALUSrc now controls whether to write to rd or rt, here it will write to "rd", actually the first 5 bits of Immediate, which is 00111. Thus the instruction writes to $7. OPR2 input of the ALU will choose RD2 instead of the immediate, because RegDst = 0. Thus $7 will receive ori $8, $7, giving us 1000 or 0111 = 1111. Thus $7 gets the value 15.

sll $10, $10, 1 $10 will contain ***41088*** `after execution.

Check:
sll $10, $10, 1
opcode = 000000
rs = 000000 (not used)
rt = $10 = 01010
rd = $10 = 01010
shamt = 00001
func = 000000
Total instruction = 000000 01010 01010 01010 00001 000000 (blue part shows the last 16 bits)
For sll, RegDst = 1 and ALUSrc = 0. Since ALUSrc now controls whether to write to rd or rt, here it will write to "rt", which is $10. OPR2 input of the ALU will choose the immediate, because RegDst = 1. Thus $10 = 16 bit immediate value shifted by 1 bit to the left.

The 16 bit "immediate" (actually rd, shamt and func) is 0101 0000 0100 0000. Shifted left 1 gives us: 1010 0000 1000 0000. Note here the left-most bit is NOT -32768 because this is now a 32-bit number (the first 16 bits are just ommitted for brevity) due to sign extension of the original immediate value. This is thus 32768 + 8192 + 128 = 41088