CS2100 MID-TERM TEST ANSWER SHEET

AY2018/19 Semester 2

TOTAL SCORE

/ 40

STUDENT NO.: | A | 0 | 1 | | | | | | |

NOTE: Write your particulars above legibly using **PEN**.

TUTORIAL GROUP:

[2 marks/MCQ]

1.	2.	3.	4.	5.	6.
C	A	C	E	B	D

7. [8 marks]

C Code	MIPS Code

`int main(void) {`
`  i = 0;`
`  A = 'A';`
`  a = 'a';`
`  Z = 'Z';`
	`[ addi $s1, $zero, 0 ]`
	`[ addi $s3, $zero, 65 ]`
	`[ addi $s4, $zero, 97 ]`
	`[ addi $s5, $zero, 90 ]`

`  do {`	`loop:`

`    if( str[i] >= 'A'`
`        && str[i] <= 'Z' ) {`
	`add $s7, $s0 , $s1`
	`lb $t2, 0($s7)`
	`slt $t1, $t2 , $s3`
	`bne $t1, $zero, else`
	`[ slt $t1, $s5 , $t2 ]`
	`[ beq $t1, $zero, else ]`

`      func(str + i);`	`j func` `ret:`
`    }`	`else:`
`[   i++   ];`	`addi $s1, $s1, 1`
`  } while(str[i-1] != 0);`	`[ bne $t2, $zero, loop ]`
`  return 0;` `}`	`quit: j exit`
// Code omitted	*# Code omitted*
`void func(char* str) {`	`func:`

`[ *str = *str + 'a' – 'A'   ];`
`  return;`
	`lb $t8, 0($s7)`
	`addi $t8, $t8, 97`
	`addi $t8, $t8, -65`
	`sb $t8, 0($s7)`
	`j ret`

`}`	
// Code omitted	*# Code omitted*
// End of Program	*exit:*

Please turn over…

8. [2 marks]

```
cs2100 is easy!
```

9. [2 marks]

```
0x15200003
```

10. **[CHALLENGING]** [2 marks]

$2^{26} - 9$ instructions [67,108,855 instructions]

11. [6 marks]

```
-5.84375
```

12. [6 marks]

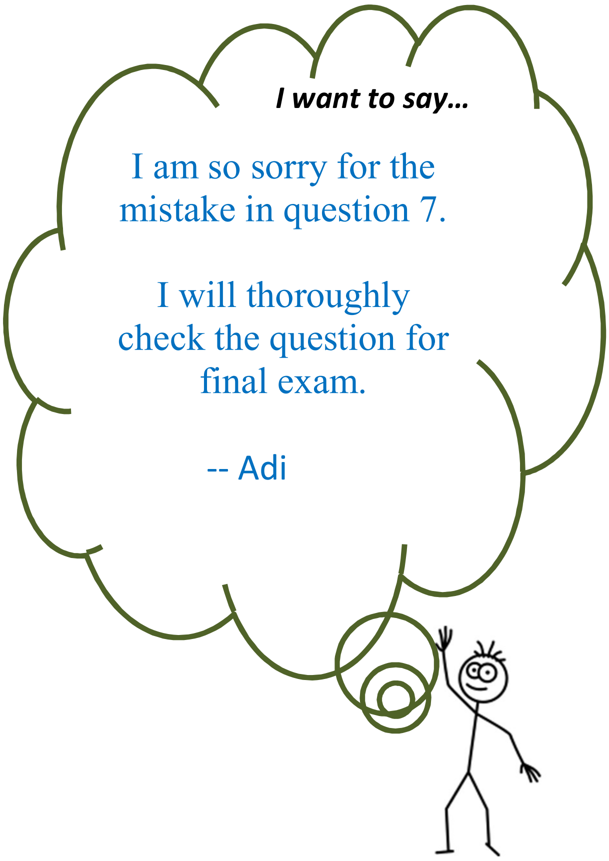
Registers File				ALU		Data Memory	
RR1	RR2	WR	WD	Opr1	Opr2	Address	Write Data
$8	$0	$31 / $ra	M[R[$8]-R[$0]] / M[R[$8]]	R[$8]	R[$0] / 0	R[$8]-R[$0] / R[$8]	R[$0] / 0

13. **[CHALLENGING]** [2 marks]

```
sw $zero, 0($zero) [or sw X, 0($zero)]
```

REFLECTIONS

Any thought about the module?
Share it in the thought bubble on the right.
This will not be graded! ☺



WORKINGS:

1. By trial and error:
   - $32_8 = 26_{10} = 2 \times 13$
   - $32_9 = 29_{10} = 1 \times 29$ (this is a prime number, not a product of two primes)
   - $32_{11} = 35_{10} = 5 \times 7$

   NOTE: prime is still prime in any base. Note, 1 is NOT prime!
2. By tracing the value of **`$t0`**, **`$t1`**, and **`$t2`** line by line

Instruction	`$t0`	`$t1`	`$t2`
`lui $t0, 0xAAAA`	0xAAAA0000	-	-
`srl $t0, $t0, 16`	0x0000AAAA	-	-
`lui $t0, 0xA0A0`	***0xA0A00000***	-	-
`ori $t1, $zero, 0x5555`	0xA0A00000	0x00005555	-
`and $t2, $t1 , $t0`	0xA0A00000	0x00005555	0x00000000

For Question 3-4, first compute the required number of bits:

- **Registers:** there are *6 registers*, therefore **3-bits**
- **Addresses:** there are *64 addresses*, therefore **6-bits**
- For Class A:
  - 3 registers: 9-bits
  - Opcode: 16-9 = ***7-bits***
- For Class B:
  - 1 address: 6-bits
  - 2 registers: 6-bits
  - Opcode: 16-6-6 = ***4-bits***

Draw a possible bit arrangement

	4-bits	3-bits	3-bits	3-bits	3-bits
Class A	Opcode$_1$	Reg_1	Reg_2	Reg_3	Opcode$_2$

	4-bits	3-bits	3-bits	6-bits
Class B	Opcode$_1$	Reg_1	Reg_2	Address

3. Maximum is achieved by maximising Class A instruction (hence, minimising Class B):
   - Assign only 1 Opcode_1 to Class B: *for instance, 0000 is for Class B*
   - Assign the rest to Class A:
     - Opcode$_1$: 0001 to 1111: $2^4 - 1 = 15$
     - Opcode$_2$: $2^3 = 8$
     - Total: $15 \times 8 = 120$
   - Sum for both classes: $120 + 1 = 121$
4. Minimum is achieved by maximising Class B instruction (hence, minimising class A):
   - Assign only 1 Opcode_1 to Class A: *for instance, 0000 is for Class A*
     - Opcode$_1$: 0000 only: 1
     - Opcode$_2$: $2^3 = 8$
     - Total: $1 \times 8 = 8$
   - Assign the rest to Class B:
     - Opcode$_1$: 0001 to 1111: $2^4 - 1 = 15$
     - Total: 15
   - Sum for both classes: $8 + 15 = 23$

5. **NOTE:** Function call is *pass-by-value*. Therefore, the array `numer` inside `rational` is *copied*.
6. **NOTE:** On the other hand, we are passing pointers directly here. Hence, *pass-by-reference*.

7. **IDEA:**
   - ***First 4 MIPS code:*** Refer to the ASCII table for the decimal value.
   - ***Next 2 MIPS code:*** Perform **`slt`** followed by either **bne** or **beq** similar to the two lines above this.
     - **`str[i] <= 'Z'`**
     - **`$t2 <= $s5`**
     - **`!($s5 < $t2)`**
     - **`($s5 < $t2) == 0`**
     - **`slt $t1, $s5, $t2 ;; bne $t1, $zero, else`**
   - ***First 1 C code:*** This is simply **`$s1++`** (or **`i++`**) after mapping.
   - ***Next 1 MIPS code:*** Simply **`bne $t2, $zero, loop`** (cannot use **`j loop`** here as **`j`** is unconditional). NOTE: since there is an **`i++`** before, **`str[i-1]`** is exactly **`$t2`**.
   - ***Last 1 C code:*** Start by translating the 4 MIPS code
     - `lb   $t8, 0($s7)   => $t8 = *str`
     - `addi $t8, $t8,  97 => $t8 = $t8 + 'a'  => $t8  = *str + 'a'`
     - `addi $t8, $t8, -65 => $t8 = $t8 – 'A'  => $t8  = *str + 'a' – 'A'`
     - `sb   $t8, 0($s7)   => *str = $t8       => *str = *str + 'a' – 'A'`
8. **TRACE:** The program basically converts uppercase to lowercase: **`"cs2100 is easy!"`**
9. **STEPS:**
   - Compute immediate value: **`bne $t1, $zero, 3`**
   - Compute registers value: **`bne $9, $0, 3`**
     - opcode: 000101
     - rs: 01001
     - rt: 00000
     - immediate: 0000 0000 0000 0011
   - Binary: 0001 0101 0010 0000 0000 0000 0000 0011
   - Hexadecimal: 1 5 2 0 0 0 0 3
   - Answer: **`0x15200003`**
10. **STEPS:**
    - Compute maximum jump using `j` instruction
      - **`j func:`** between `ret` label and `func` label, need to be within 256MB boundary
      - **`j exit:`** between top #Code `omitted` and `exit` label, need to be within 256MB boundary
      - **`j ret :`** between bottom #Code `omitted` and `ret` label, need to be within 256 MB boundary
      - Due to overlap between these `j` instructions, it implies between `ret` label and `exit` label are within 256MB boundary
    - 256MB boundaries contain 2^{26} instructions.
    - Subtract 8 instructions already in the region.
    - Subtract 1 since we must jump to `exit` label, which means there must be one instruction there at the `exit` label. Note that the label is outside the #Code `omitted` region.
    - Total: $2^{26} - 9$

11. **STEPS:**
    - Convert to binary: `1100 0000 1011 1011 0000 0000 0000 0000`
    - Split into region:
        - Sign: 1 (*negative*)
        - Exponent: 10000001
            - Convert to decimal: 129
            - Excess-127: 129 = 127+2 => 2
        - Mantissa: 01110110000000000000000
            - Normalize: 1.0111011×2^2
            - Remove exponent: 101.11011
            - Convert to decimal: $5 + 2^{-1} + 2^{-2} + 2^{-4} + 2^{-5}$
            - Sum: $5 + 0.5 + 0.25 + 0.0625 + 0.03125 = 5.84375$
    - Answer: **`-5.8435`**

12. **TRACE:** Note that the Control signal is different. Be careful with which values are selected in multiplexer as well as the behaviour of each component (e.g., Register File and Data Memory). WR is the first 5 bits, so it will be **`$31`** (note the use of **`$`** sign to indicate register).
13. **NOTE:** Need to ensure that ALU operation results in **`ALUresult = 0`** to force **`is0? = 1`**. This can be guaranteed by having **`sw $??, 0($zero)`**. Note that **`$??`** can be any register.
    This is a branch, since the **`PCSrc`** is set to 1, *but it branch to the next instruction*.