w

United International University
Department of CSE
CSE 313: Computer Architecture
Midterm Examination
Spring 2022

Time: 1 hour and 45 minutes **Full Marks: 30**

[Any examinee found adopting unfair means will be expelled from the trimester / program as per UIU disciplinary rules.]

[N.B.: Answer all the questions. Assume any data if it is not mentioned explicitly.]

1. a) For the same program, two different compilers are used. The table below shows the execution time of the two different compiled programs. [4]

	Compiler A		Compiler B	
	Number of instructions	Execution Time	Number of instructions	Execution Time
Program 1	$1x10^9$	1s	$1.2x10^9$	1.4s
Program 2	$1x10^9$	0.8s	$1.2x10^9$	0.7s

i) Find the average CPI for each program given that the processor has a clock cycle time of 1ns.

ii) Use the average CPIs found in part (i), but that the compiled programs run on two different processors. If the execution times on the two processors are the same (5s), how much faster is the clock of the processor running compiler A's code versus the clock of the processor running compiler B's code?

b) Suppose, there are three classes of instructions A, B, C in a particular instruction set architecture with CPIs 1.2, 2 and 2.5 respectively. The number of instructions from each class in two separate programs are as follows: [4]

Programs	Instruction classes		
	A	B	C
P1	40	10	16
P2	12	13	40

If the clock frequency is 2GHz, then what is the total execution time for P1? If the total time for P1 is 100 ns, then find out the improvement factor to make it two times faster.

c) What is the power wall? Explain how the introduction of multi core processors has overcome power limitation. [2]

2. Consider the following C function that calculates the sum of an array, *arr*, from index *low* to index *high*. Given that the base address of *arr* is contained in register *$s0*, the variable *i* and *sum* is contained in register *$s1* and *$s2* respectively, and the starting MIPS assembly instruction address is 1000.

```
int func(int low , int high)
{
    int sum = 0,i;
    for(i = low ; i < high ; i++)
    {
        sum = sum + arr[i];
    }
    return sum;
}
```

a) Convert the code to the corresponding MIPS assembly instructions [6]

b) Convert **the first 15 lines** of your assembly instructions to corresponding machine code. No need to convert it to binary. [5]

c) A processor architect makes the following claim "The maximum array size that MIPS can handle is 32, as there are only 32 possible registers in MIPS to hold data." Briefly describe how arrays are stored in MIPS. In your answer, include the instructions that are used to access arrays. Based on this, explain if the architect's claim is correct. Also, find out the **maximum array index size** we can have in the MIPS architecture. [4]

3. a) Assuming 4 bit architecture and using the division algorithm show each step of the division of 13 by 6. [3]

b) Optimized multiplication is better than the normal multiplication algorithm. Why? Explain. [2]

MIPS Machine Codes

Instruction	Opcode	Function Code
add	0	32
sub	0	34
lw	35	
sw	43	
and	0	36
or	0	37
nor	0	39
andi	12	
ori	13	
sll	0	0
srl	0	2
beq	4	
bne	5	
slt	0	42
j	2	
jr	0	8
jal	3	
addi	8	

MIPS Registers

Name	Register Number
$zero	0
$at	1
$v0-$v1	2-3
$a0-$a3	4-7
$t0-$t7	8-15
$s0-$s7	16-23
$t8-$t9	24-25
$k0-$k1	26-27
$gp	28
$sp	29
$fp	30
$ra	31

United International University

Name [redacted]

ID [redacted]

Invigilator's Signature with date

Course Code CSE 3313 Course Title Computer Architecture Section [redacted]

Semester / Trimester : Spring / Summer / Fall, 20 2 2

Name of Exam : Mid-term 1 / Mid-term 2 / Final Date of Examination : 2nd April, Saturday /2022

Question	Marks
01	
02	
03	
04	
05	
06	
07	
08	
09	
10	
11	
12	
13	
14	
15	
Total	25

Rules

- Any examinee found adopting unfair means will be expelled from the Trimester / Program as per UIU Disciplinary Rules.
- Unauthorized possession / use of a calculator, digital diary, mobile phone and other electronic data storage devices are strictly prohibited inside the examination hall.
- No examinee is allowed to go outside the examination hall before completing the examination. Invigilator can give permission in an exceptional / emergency case.
- Student must write down ID No. on the additional answer script(s), if any; and attach additional answer script(s) with the main script within the duration of examination. Make sure that the additional answer script(s) is/ are signed by the invigilator.
- An examinee must hand over the answer script and unused additional answer sheet (s) if any, to the invigilator before leaving the examination hall.
- An examinee should not leave the examination hall until all answer scripts are collected and the invigilator asks him/her to leave the examination hall.

.....

Signature of the Examiner

1. a) For Compiler A

i) For Program 1

Number of cycles in A $= \frac{1}{1\times10^{-9}} = 10^9$

Number of cycles in B $= \frac{1.4}{1\times10^{-9}} = 1.4\times10^9$

Average CPI $= \frac{\text{Total number of cycles}}{\text{Total number of instructions}}$

$= \frac{10^9 + 1.4\times10^9}{1\times10^9 + 1.2\times10^9}$

$= \frac{2.4\times10^9}{2.2\times10^9} = \boxed{1.09}$

For Program 2

Number of cycles in A $= \frac{0.8}{1\times10^{-9}} = 8\times10^8$

Number of cycles in B $= \frac{0.7}{1\times10^{-9}} = 7\times10^8$

Average CPI $= \frac{\text{Total number of cycles}}{\text{Total number of instructions}}$

$= \frac{8\times10^8 + 7\times10^8}{1\times10^9 + 1.2\times10^9}$

$= \frac{1.5\times10^9}{2.2\times10^9} = \boxed{0.68}$

ii) Taking Program 1

$\text{Clock Rate}_A = \frac{CPI_A \times IC_A}{\text{CPU Time}_A} = \frac{1.09 \times 1\times10^9}{5} = 218\ \text{MHz}$

$\text{Clock Rate}_B = \frac{CPI_B \times IC_B}{\text{CPU Time}_B} = \frac{1.09 \times 1.2\times10^9}{5} = 261.6\ \text{MHz}$

1. b) ~~CPU Time~~ $_{P1}$ = c

$$\text{Average CPI}_{P1} = \frac{(1.2 \times 40) + (2 \times 10) + (2.5 \times 16)}{66} = \frac{18}{11}$$

$$\text{CPU Time}_{P1} = \frac{\text{Average CPI}_{P1} \times \text{IC}_{P1}}{\text{Clock Rate}_{P1}} = \frac{\frac{18}{11} \times 66}{2 \times 10^9} = \boxed{5.4 \times 10^{-8}\ s}$$

$$\text{Overall improvement factor} = \frac{T_{original}}{T_{improved}}$$

$$\Rightarrow 2 = \frac{100 \times 10^{-9}}{5.4 \times 10^{-8}\ T_{improved}}$$

$$\Rightarrow T_{improved} = 50 \times 10^{-9}$$

$$T_{improved} = \frac{T_{affected}}{n} + T_{unaffected}$$

$$\Rightarrow 50 \times 10^{-9} = \frac{5.4 \times 10^{-8}}{n} + (100 \times 10^{-9} - 5.4 \times 10^{-8})$$

$$\Rightarrow \boxed{\text{Improvement Factor, } n = 13.5}$$

Q.

c) Ideally, we would like to increase frequency ~~and~~ while keeping the power low. This is done by increasing the frequency and decreasing voltage, as $P = CV^2f$. So, heat energy production is kept low as power is low. But we get to a point where reducing the voltage is no longer viable. The processor would start behaving inefficiently if we reduced the voltage any further. At that point, there is no way to increase the frequency without increasing the power. This is known as the power wall, which is a constraint in uniprocessors.

With the introduction of multi core processors, which are multiple processors on the same chip, the frequency can be increased further, as the processors divide the task among themselves and ~~solve~~ carry out the computations individually, before accumulating everything ~~before~~ to ~~sho~~ produce the output. This is known as instruction-level parallelism. Each individual processor has low power but altogether, they have high frequency.

2. a) Assembly Code

```
1000  Func:  addi  $sp, $sp, -12
1004         sw    $s0, 8($sp)
1008         sw    $s1, 4($sp)
1012         sw    $s2, 0($sp)

1016         addi  $s2, $zero, $zero   # sum = 0
1020         add   $s1, $a0, $zero     # i = low
1024  Loop:  slt   $t0, $s1, $a1       # $t0 = 1 if i < high, else $t0 = 0
1028         beq   $t0, $zero, Break   # break if $t0 = 0   (1056)
1032         sll   $t1, $s1, 2         # $t1 = i x 4 = offset address
1036         add   $t1, $s0, $t1       # $t1 = base + offset = exact address
1040         lw    $t2, 0($t1)         #
1044         add   $s2, $s2, $t2       # sum = sum + arr[i]
1048         addi  $s1, $s1, 1         # i++
1052         j     Loop (1024)
1056  Break: add   $v0, $s2, $zero
1060         lw    $s2, 0($sp)
1064         lw    $s1, 4($sp)
1068         lw    $s0, 8($sp)
1072         addi  $sp, $sp, 12
1076         jr    $ra
```

b) Machine Code

	op	rs	rt	rd	shamt	funct	C or A
1000	8	29	29				-12
1004	43	29	16				8
1008	43	29	17				4
1012	43	29	18				0
1016	0	0	0	18	x	32	
1020	0	4	0	17	x	32	
1024	0	17	5	8	x	42	
1028	4	0	8				264
1032	0	x	17	9	2	0	
1036	0	16	9	9	x	32	
1040	35	9	10				0
1044	0	18	10	18	x	32	
1048	8	17	17				1
1052	2						256
1056	0	18	0	2	x	32	
1060	35	29	18				0
1064							
1068							
1072							
1076							

format!

6.

4.5

2. c) In MIPS, when an array is declared, a base address is set for the array. We use 'sw' ~~to store~~ or store word to store data in the array. For example, sw $sl, 4($sp) stores the value in register $sl to the $\frac{4}{4} = 1^{st}$ index of the array represented by the base address set in the register $sp.

We can also load word from the array to a register by using 'lw'. For example, lw $sl, 4($sp) ~~to~~ saves the value of the $\frac{4}{4}=1^{st}$ index of the array to the register $sl.

The architect's claim is correct, since a single register ~~can store~~ has 32 bits. ✗

2

3. a) 13 → 1101

6 → 0110

1101 ÷ 0110

Iteration 1

Divisor : 0110 0000
Quotient: 0000
Remainder: 0000 1101
Remainder − Divisor = negative
(Remainder Unchanged)
Divisor: 00110000 (RS)
Quotient: 0000 (LS)
Is i = 5 ? → No.

Iteration 2

Divisor: 00110000
Quotient: 0000
Remainder: 00001101
Remainder − Divisor = negative
(Remainder unchanged)
Divisor: 00011000 (RS)
Quotient: 0000 (LS)
Is i = 5 ? → No.

Iteration 3

Divisor: 00011000
Quotient: 0000
Remainder: 00001101
Rem. − Divisor = negative
(Rem. unchanged)
Divisor: 00001100 (RS)
Quotient: 0000 (LS)
Is i=5 ? → No.

Iteration 4

Divisor: 00001100
Quotient: 0000
Remainder : 00001101
Rem. − Divisisor = positive
Remainder = Rem. − Divisor = 00001101 − 00001100 = 00000001
Divisor : 00000110 (RS)
Quotient: 0001 (LS)
Is i=5? → No.

Iteration 5

Divisor : 00000110
Quotient: 0001
Remainder : 00000001
Rem. − Divisor = negative
(Rem. unchanged)
Divisor: 00000011 (RS)
Quotient: 0010 (LS)
Is i=5 ? → Yes
Done.

3. b) 1) In normal multiplication, 3 registers are used, Multiplicand, Multiplier and Product.
But in optimized multiplication, 2 registers are used, Multiplicand and Product.
(The product register itself stores the multiplier as well as the product).

2) Optimized multiplication also uses the Multiplicand register with 32 bits, which is less than the 64 bits the normal multiplication uses.