Name Surname:
ID:

Q1 (35pts)	Q2 (35pts)	Q3 (35pts)	Total (105pts)

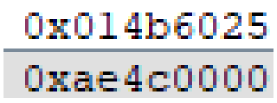
CSE 331/503 Quiz 1 – Fall 2021

1. Assume base address of integer array **`a[]`** is at **`$s0`** and use **`$s1`** for **`i`**. Convert the following C code segment into assembly:

```
i = 0;

while(i < 71){

   a[i] = a[i] & a[i + 8];

   ++i;

   };
```

```
                Addi $s1, $zero, 0     #i=0
                Addi $t0 $zero, 71

whileLoop:      Beq $s1, $t0, exit
                sll $t1, $s1, 2        #4*i
                add $t2, $t1, $s0      #s0+4*i
                lw $t3, 0($t2)         #a[i]
                lw $t4, 32($t2)        #a[i+8]
                and $t3, $t3, $t4      # a[i] = a[i] & a[i+8]
                sw $t3, 0($t2)
                addi $s1, $s1, 1       #++i
                j whileLoop

exit:           addi $v0, $zero, 10
                syscall
```

2. Convert below two instructions in machine code to assembly. Use register names instead of register addresses in your assembly. (For instance use $t0 instead of $8).



```
0x014b6025
0xae4c0000
```

Binary	= 0000 0001 0100 1011 0110 0000 0010 0101					
Instruction format =	000000	01010	01011	01100	00000	10 0101
	Opcode	Rs	Rt	Rd	shamt	funct
	0 = R type	10 = t2	11=t3	12=t4	0	25 (hex)

Answer = or $t4, $t2, $t3

Binary	= 1010 1110 0100 1100 0000 0000 0000 0000			
Instruction format =	101011	10010	01100	0000 0000 0000 0000
	Opcode	Rs	Rt	imm
	2b = I type	18= s2	12=t4	0

Answer = sw $t4, 0($s2)

3. You want to improve the execution time of your processor by 10% for a specific program. The program has the following time distribution among different operations:

Type of operation (Hardware segment)	Percentage of time
Multiplication	20%
Division	10%
Addition	30%
Load/store	25%
Others	5%

The amount of time (in terms of days) to speed up any hardware segment is given by the following equation:

$$t = 30n^2 + 80n + 100$$

where n is the speed up amount for any operation. For instance for 2 times speed up the required development period is $30\text{x}4 + 80 \text{ x } 2 + 100 = 380$ days.

Explain which operation (hardware segment) must be improved and by how many times so that the resultant speed up of the program is 10% and the required time period is the least one. Also compute and write the amount of time (in terms of days) required for that speed up.

Initial case = 100x time , final case = after the 1.1 speed up, 90.9x = 90x time

We focus on the most used opeartions like addition(30%), after then load/store(25%). We choose the most used that is addition operation, because of the decrease the amount of day to speed up.

No matter which operation we choose, we want 10x time to be decreased. Addition operation improved 20x time instead of 30x time. Thus, we obtained 90x total execution time.

For addition operation, we need 30x/20x = 1.5 times speed up

$\mathrm{T} = 30\text{x}2.25 + 80 \text{ x } 1.5 + 100 = 287.5 \text{ days}$

REGISTERS

NAME	NMBR	USE
$zero	0	The Constant Value 0
$at	1	Assembler Temporary
$v0-$v1	2-3	Values for Function Results and Expression Evaluation
$a0-$a3	4-7	Arguments
$t0-$t7	8-15	Temporaries
$s0-$s7	16-23	Saved Temporaries
$t8-$t9	24-25	Temporaries
$k0-$k1	26-27	Reserved for OS Kernel
$gp	28	Global Pointer
$sp	29	Stack Pointer
$fp	30	Frame Pointer
$ra	31	Return Address
$f0-$f31	0-31	Floating Point Registers

31 26	21	16	11	6	0
op	**rs**	**rt**	**rd**	**shamt**	**funct**
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

31 26	21	16	0
op	**rs**	**rt**	**immediate**
6 bits	5 bits	5 bits	16 bits

31 26	0
op	**target address**
6 bits	26 bits

CORE INSTRUCTION SET (INCLUDING PSEUDO INSTRUCTIONS)

NAME	MNE-MON-IC	FOR-MAT	OPERATION (in Verilog)		OPCODE/ FUNCT (Hex)
Add	add	R	R[rd]=R[rs]+R[rt]	(1)	0/20
Add Immediate	addi	I	R[rt]=R[rs]+SignExtImm	(1)(2)	8
Add Imm. Unsigned	addiu	I	R[rt]=R[rs]+SignExtImm	(2)	9
Add Unsigned	addu	R	R[rd]=R[rs]+R[rt]	(2)	0/21
Subtract	sub	R	R[rd]=R[rs]-R[rt]	(1)	0/22
Subtract Unsigned	subu	R	R[rd]=R[rs]-R[rt]		0/23
And	and	R	R[rd]=R[rs]&R[rt]		0/24
And Immediate	andi	I	R[rt]=R[rs]&ZeroExtImm	(3)	c
Nor	nor	R	R[rd]=∼(R[rs]|R[rt])		0/27
Or	or	R	R[rd]=R[rs]|R[rt]		0/25
Or Immediate	ori	I	R[rt]=R[rs]|ZeroExtImm	(3)	d
Xor	xor	R	R[rd]=R[rs]ˆR[rt]		0/26
Xor Immediate	xori	I	R[rt]=R[rs]ˆZeroExtImm		e
Shift Left Logical	sll	R	R[rd]=R[rs]≪shamt		0/00
Shift Right Logical	srl	R	R[rd]=R[rs]≫shamt		0/02
Shift Right Arithmetic	sra	R	R[rd]=R[rs]≫>shamt		0/03
Shift Left Logical Var.	sllv	R	R[rd]=R[rs]≪R[rt]		0/04
Shift Right Logical Var.	srlv	R	R[rd]=R[rs]≫R[rt]		0/06
Shift Right Arithmetic Var.	srav	R	R[rd]=R[rs]≫>R[rt]		0/07
Set Less Than	slt	R	R[rd]=(R[rs]<R[rt])?1:0		0/2a
Set Less Than Imm.	slti	I	R[rt]=(R[rs]<SignExtImm)?1:0	(2)	a
Set Less Than Imm. Unsign.	sltiu	I	R[rt]=(R[rs]<SignExtImm)?1:0	(2)(6)	b
Set Less Than Unsigned	sltu	R	R[rd]=(R[rs]<R[rt])?1:0	(6)	0/2b
Branch On Equal	beq	I	if(R[rs]==R[rt]) PC=PC+4+BranchAddr	(4)	4
Branch On Not Equal	bne	I	if(R[rs]!=R[rt]) PC=PC+4+BranchAddr	(4)	5
Branch Less Than	blt	P	if(R[rs]<R[rt]) PC=PC+4+BranchAddr		
Branch Greater Than	bgt	P	if(R[rs]>R[rt]) PC=PC+4+BranchAddr		
Branch Less Than Or Equal	ble	P	if(R[rs]<=R[rt]) PC=PC+4+BranchAddr		
Branch Greater Than Or Equal	bge	P	if(R[rs]>=R[rt]) PC=PC+4+BranchAddr		
Jump	j	J	PC=JumpAddr	(5)	2
Jump And Link	jal	J	R[31]=PC+4; PC=JumpAddr	(5)	2
Jump Register	jr	R	PC=R[rs]		0/08
Jump And Link Register	jalr	R	R[31]=PC+4; PC=R[rs]		0/09
Move	move	P	R[rd]=R[rs]		
Load Byte	lb	I	R[rt]={24’b0, M[R[rs]+ZeroExtImm](7:0)}	(3)	20
Load Byte Unsigned	lbu	I	R[rt]={24’b0, M[R[rs]+SignExtImm](7:0)}	(2)	24
Load Halfword	lh	I	R[rt]={16’b0, M[R[rs]+ZeroExtImm](15:0)}	(3)	25
Load Halfword Unsigned	lhu	I	R[rt]={16’b0, M[R[rs]+SignExtImm](15:0)}	(2)	25
Load Upper Imm.	lui	I	R[rt]={imm,16’b0}		f
Load Word	lw	I	R[rt]=M[R[rs]+SignExtImm]	(2)	23
Load Immediate	li	P	R[rd]=immediate		
Load Address	la	P	R[rd]=immediate		
Store Byte	sb	I	M[R[rs]+SignExtImm] (7:0)=R[rt](7:0)	(2)	28
Store Halfword	sh	I	M[R[rs]+SignExtImm] (15:0)=R[rt](15:0)	(2)	29
Store Word	sw	I	M[R[rs]+SignExtImm]=R[rt]	(2)	2b