VE370 RC1

Exercise 1.3

Consider three different processors P1, P2, and P3 executing the same instruction set with the clock rates and CPIs given in the following table.

	Processor	Clock Rate	CPI
a.	P1	3 GHz	1.5
	P2	2.5 GHz	1.0
	P3	4 GHz	2.2
b.	P1	2 GHz	1.2
	P2	3 GHz	0.8
	P3	4 GHz	2.0

1.3.1 [5] <1.4> Which processor has the highest performance expressed in instructions per second?

1.3.2 [10] <1.4> If the processors each execute a program in 10 seconds, find the number of cycles and the number of instructions.

1.3.3 [10] <1.4> We are trying to reduce the time by 30% but this leads to an increase of 20% in the CPI. What clock rate should we have to get this time reduction?

$$\text{CPU Time} = \text{CPU Clock Cycles per program} \times \text{Clock Cycle Time}$$

$$= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}$$

$$\text{Clock Cycles} = \text{Instruction Count} \times \text{Clock Cycles per Instruction (CPI)}$$

$$\text{CPU Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

$$= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$

Classic CPU performance equation

- 1.3.1
  - $\textbf{CPU Time} = \frac{\textbf{Instruction Count} * \textbf{CPI}}{\textbf{Clock Rate}}$
- 1.3.2
  - $\textbf{\# of Cycles} = \textbf{time} * \textbf{Clock Rate}$
  - $\textbf{\# of Instruction} = \frac{\textbf{\# of Cycles}}{\textbf{CPI}}$
- 1.3.3
  - $\textbf{Clock Rate}' = \frac{\textbf{Instruction Count} * \textbf{CPI} * 1.2}{\textbf{CPU Time} * 0.7} = 1.71 \textbf{Clock Rate}$

Solution	a			b		
Processor	P1	P2	P3	P1	P2	P3
CPU Time	0.5	0.4	0.55	0.6	0.27	0.5
# of cycles (10^{10})	3	2.5	4	2	3	4
# of instruction (10^{10})	2	2.5	1.8	1.67	3.75	2

The following table shows the number of instructions for a program.

	Arith	Store	Load	Branch	Total
a.	650	100	600	50	1400
b.	750	250	500	500	2000

1.4.4 [5] <1.4> Assuming that arith instructions take 1 cycle, load and store 5 cycles, and branches 2 cycles, what is the execution time of the program in a 2 GHz processor?

1.4.5 [5] <1.4> Find the CPI for the program.

1.4.6 [10] <1.4> If the number of load instructions can be reduced by one half, what is the speedup and the CPI?

- 1.4.4
  - $\textbf{CPU Time} = \frac{\textbf{Instruction Count} * \textbf{CPI}}{\textbf{Clock Rate}}$
  - (a): $\textbf{CPU Time} = \frac{650*1+100*5+600*5+50*2}{2*10^9} = 2125\text{ns}$
  - (b): $\textbf{CPU Time} = \frac{750*1+250*5+500*5+500*2}{2*10^9} = 2750\text{ns}$

The following table shows the number of instructions for a program.

	Arith	Store	Load	Branch	Total
a.	650	100	600	50	1400
b.	750	250	500	500	2000

1.4.4 [5] <1.4> Assuming that arith instructions take 1 cycle, load and store 5 cycles, and branches 2 cycles, what is the execution time of the program in a 2 GHz processor?

1.4.5 [5] <1.4> Find the CPI for the program.

1.4.6 [10] <1.4> If the number of load instructions can be reduced by one half, what is the speedup and the CPI?

- 1.4.5
  - $\mathbf{CPI} = \frac{\mathbf{Tme} * \mathbf{Clock\ Rate}}{\mathbf{Instruction\ Count}}$
  - (a): $\mathbf{CPI} = \frac{\mathbf{2125ns} * \mathbf{2GHz}}{\mathbf{1400}} = \mathbf{3.03}$
  - (b): $\mathbf{CPI} = \frac{\mathbf{2750ns} * \mathbf{2GHz}}{\mathbf{2000}} = \mathbf{2.75}$

- 1.4.6
  - (a):
    - $\mathbf{CPU\ Time} = \frac{650*1+100*5+300*5+50*2}{2*10^9} = \mathbf{1375ns}$
    - $\mathbf{Speed-up} = \frac{2125}{1375}$
    - $\mathbf{CPI} = \frac{1375ns*2GHz}{1400} = \mathbf{1.96}$
  - (b):
  - $\mathbf{CPU\ Time} = \frac{750*1+250*5+250*5+500*2}{2*10^9} = \mathbf{2125ns}$
    - $\mathbf{Speed-up} = \frac{2750}{2125}$
    - $\mathbf{CPI} = \frac{2125ns*2GHz}{2000} = \mathbf{2.125}$

The table below shows the instruction type breakdown of a given application executed on 1, 2, 4, or 8 processors. Using this data, you will be exploring the speed-up of applications on parallel processors.

	Processors	No. Instructions per Processor			CPI		
		Arithmetic	Load/Store	Branch	Arithmetic	Load/Store	Branch
a.	1	2560	1280	256	1	4	2
	2	1280	640	128	1	5	2
	4	640	320	64	1	7	2
	8	320	160	32	1	12	2

	Processors	No. Instructions per Processor			CPI		
		Arithmetic	Load/Store	Branch	Arithmetic	Load/Store	Branch
b.	1	2560	1280	256	1	4	2
	2	1280	640	128	1	6	2
	4	640	320	64	1	8	2
	8	320	160	32	1	10	2

1.10.1 [5] <1.4, 1.6> The table above shows the number of instructions required per processor to complete a program on a multiprocessor with 1, 2, 4, or 8 processors. What is the total number of instructions executed per processor? What is the aggregate number of instructions executed across all processors?

1.10.2 [5] <1.4, 1.6> Given the CPI values on the right of the table above, find the total execution time for this program on 1, 2, 4, and 8 processors. Assume that each processor has a 2 GHz clock frequency.

1.10.3 [10] <1.4, 1.6> If the CPI of the arithmetic instructions was doubled, what would the impact be on the execution time of the program on 1, 2, 4, or 8 processors?

- 1.10.1
  - Take the 8 processors as an example
  - # of instructions per processors=320+160+32=512
  - Total number of instructions=512*8

processor	# of instructions per processors	Aggregate # of instructions
1	4096	4096
2	2048	4096
4	1024	4096
8	512	4096

The table below shows the instruction type breakdown of a given application executed on 1, 2, 4, or 8 processors. Using this data, you will be exploring the speed-up of applications on parallel processors.

	Processors	No. Instructions per Processor			CPI		
		Arithmetic	Load/Store	Branch	Arithmetic	Load/Store	Branch
a.	1	2560	1280	256	1	4	2
	2	1280	640	128	1	5	2
	4	640	320	64	1	7	2
	8	320	160	32	1	12	2

	Processors	No. Instructions per Processor			CPI		
		Arithmetic	Load/Store	Branch	Arithmetic	Load/Store	Branch
b.	1	2560	1280	256	1	4	2
	2	1280	640	128	1	6	2
	4	640	320	64	1	8	2
	8	320	160	32	1	10	2

1.10.1 [5] <1.4, 1.6> The table above shows the number of instructions required per processor to complete a program on a multiprocessor with 1, 2, 4, or 8 processors. What is the total number of instructions executed per processor? What is the aggregate number of instructions executed across all processors?

1.10.2 [5] <1.4, 1.6> Given the CPI values on the right of the table above, find the total execution time for this program on 1, 2, 4, and 8 processors. Assume that each processor has a 2 GHz clock frequency.

1.10.3 [10] <1.4, 1.6> If the CPI of the arithmetic instructions was doubled, what would the impact be on the execution time of the program on 1, 2, 4, or 8 processors?

- 1.10.2
  - Take the 8 processors as an example
  - Execution time=$\frac{320+160*12+32*2}{2*10^9}$
  - Do not multiple the number of processors

a	processors	Execution time (ns)
	1	4096
	2	2368
	4	1505
	8	1152
b	processors	Execution time (ns)
	1	4096
	2	2688
	4	1664
	8	992

- 1.10.3
  - Execution time=$\frac{320*2+160*12+32*2}{2*10^9}$

The table below shows the number of instructions per processor core on a multicore processor as well as the average CPI for executing the program on 1, 2, 4, or 8 cores. Using this data, you will be exploring the speedup of applications on multicore processors.

	Cores per Processor	Instructions per Core	Average CPI
a.	1	1.00E+10	1.2
	2	5.00E+09	1.4
	4	2.50E+09	1.8
	8	1.25E+09	2.6

	Cores per Processor	Instructions per Core	Average CPI
b.	1	1.00E+10	1.0
	2	5.00E+09	1.2
	4	2.50E+09	1.4
	8	1.25E+09	1.7

1.10.4 [10] <1.4, 1.6> Assuming a 3 GHz clock frequency, what is the execution time of the program using 1, 2, 4, or 8 cores?

1.10.5 [10] <1.5, 1.6> Assume that the power consumption of a processor core can be described by the following equation:

$$\text{Power} = \frac{5.0\text{mA}}{\text{MHz}} \text{Voltage}^2$$

where the operation voltage of the processor is described by the following equation:

$$\text{Voltage} = \frac{1}{5}\text{Frequency} + 0.4$$

with the frequency measured in GHz. So, at 5 GHz, the voltage would be 1.4 V. Find the power consumption of the program executing on 1, 2, 4, and 8 cores assuming that each core is operating at a 3 GHz clock frequency. Likewise, find the power consumption of the program executing on 1, 2, 4, or 8 cores assuming that each core is operating at 500 MHz.

- 1.10.4
  - Eg. processor with 8 cores
  - Execution time=$\frac{1.25*10^9*2.6}{3*10^9}$

a	Cores per processor	Execution time (s)
	1	4
	2	2.33
	4	1.5
	8	1.08

b	Cores per processor	Execution time (s)
	1	3.3
	2	2
	4	1.17
	8	0.71

The table below shows the number of instructions per processor core on a multicore processor as well as the average CPI for executing the program on 1, 2, 4, or 8 cores. Using this data, you will be exploring the speedup of applications on multicore processors.

	Cores per Processor	Instructions per Core	Average CPI
a.	1	1.00E+10	1.2
	2	5.00E+09	1.4
	4	2.50E+09	1.8
	8	1.25E+09	2.6

	Cores per Processor	Instructions per Core	Average CPI
b.	1	1.00E+10	1.0
	2	5.00E+09	1.2
	4	2.50E+09	1.4
	8	1.25E+09	1.7

1.10.4 [10] <1.4, 1.6> Assuming a 3 GHz clock frequency, what is the execution time of the program using 1, 2, 4, or 8 cores?

1.10.5 [10] <1.5, 1.6> Assume that the power consumption of a processor core can be described by the following equation:

$$\text{Power} = \frac{5.0\text{mA}}{\text{MHz}} \text{Voltage}^2$$

where the operation voltage of the processor is described by the following equation:

$$\text{Voltage} = \frac{1}{5}\text{Frequency} + 0.4$$

with the frequency measured in GHz. So, at 5 GHz, the voltage would be 1.4 V. Find the power consumption of the program executing on 1, 2, 4, and 8 cores assuming that each core is operating at a 3 GHz clock frequency. Likewise, find the power consumption of the program executing on 1, 2, 4, or 8 cores assuming that each core is operating at 500 MHz.

- 1.10.5
  - 3GHz: Voltage=1V
  - 500MHz: Voltage=0.5V

a	Cores per processor	Power (3GHz)	Power (500MHz)
	1	5	1.25
	2	10	2.5
	4	20	5
	8	40	10

The following problems explore translating from C to MIPS. Assume that the variables `f` and `g` are given and could be considered 32-bit integers as declared in a C program.

a.	`f = -g - f;`
b.	`f = g + (-f - 5);`

2.3.1 [5] <2.2> For the C statements above, what is the corresponding MIPS assembly code? Use a minimal number of MIPS assembly instructions.

- Assume f as $s0, g as $s1
- (a)
  - sub $t0, $0, $s1 temp= -g
  - sub $s0, $t0, $s0 f= temp -g
- (b)
  - addi $t0, $s0, 5 temp=f+5
  - sub $s0, $s1, $t0 f=g-temp

The following problems deal with translating from C to MIPS. Assume that the variables f, g, h, i, and j are assigned to registers $s0, $s1, $s2, $s3, and $s4, respectively. Assume that the base address of the arrays A and B are in registers $s6 and $s7, respectively. Assume that the elements of the arrays A and B are 4-byte words:

a.	`f = f + A[2];`
b.	`B[8] = A[i] + A[j];`

2.6.1 [10] <2.2, 2.3> For the C statements above, what is the corresponding MIPS assembly code?

- (a)
  - lw $t0, 8($s6) temp=A[2] load the content of A[2]
  - add $s0, $s0, $t0 f=f+temp f=f+A[2]
- (b)
  - sll $t0, $s3, 2 temp0=i*4 get the offset of A[i]
  - add $t0, $t0, $s6 temp0+=A get the address of A[i]
  - sll $t1, $s4, 2 temp1=j*4 get the offset of A[j]
  - add $t1, $t1, $s6 temp1+=A get the address of A[j]
  - lw $t2, 0($t0) temp2=A[i] load the content of A[i]
  - lw $t3, 0($t1) temp3=A[j] load the content of A[j]
  - Add $t4, $t2, $t3 temp4=A[i]+A[j]
  - sw $t4, 32($s7) B[8]=A[i]+A[j] store the content back to B[8]

The following problems deal with translating from MIPS to C. Assume that the variables f, g, h, i, and j are assigned to registers $s0, $s1, $s2, $s3, and $s4, respectively. Assume that the base address of the arrays A and B are in registers $s6 and $s7, respectively.

a.	`sub $s0, $s0, $s1`
	`sub $s0, $s0, $s3`
	`add $s0, $s0, $s1`
		$s0=f - g;
		$s0=f – g - i;
		$s0=f - g – i + g;

b.	`addi $t0, $s6, 4`
	`add $t1, $s6, $0`
	`sw $t1, 0($t0)`
	`lw $t0, 0($t0)`
	`add $s0, $t1, $t0`
		$t0=address of A[1];
		$t1=base address of A;
		A[1]=address of A[0];
		$t0=address of A[0];
		f=base address of A + base address of A

2.6.4 [5] <2.2, 2.3> For the MIPS assembly instructions above, what is the corresponding C statement?

- (a)
  - f=f-i;
- (b)
  - f=2A;

In the following problems, the data table contains bits that represent the opcode of an instruction. You will be asked to interpret the bits as MIPS instructions into assembly code and determine what format of MIPS instruction the bits represent.

a.	$0000\ 0010\ 0001\ 0000\ 1000\ 0000\ 0010\ 0000_{two}$
b.	$0000\ 0001\ 0100\ 1011\ 0100\ 1000\ 0010\ 0010_{two}$

2.10.1 [5] <2.5> For the binary entries above, what instruction do they represent?

2.10.2 [5] <2.5> What type (I-type, R-type, J-type) instruction do the binary entries above represent?

Instruction fields

- op: operation code (opcode)
- rs: first source register number
- rt: second source register number
- rd: destination register number
- shamt: shift amount (00000 for now)
- funct: function code (extends opcode)

- 2.10.1 & 2.10.2
  - *(a)* add $s0, $s0, $s0 ——R-type

op	rs	rt	rd	shamt	func
000000	10000	10000	10000	00000	100000
0	16	16	16	0	32

  - *(b)* sub $t1, $t2, $t3 ——R-type

op	rs	rt	rd	shamt	func
000000	01010	01011	01001	00000	100010
0	10	11	9	0	34

In the following problems, the data table contains MIPS instructions. You will be asked to translate the entries into the bits of the opcode and determine the MIPS instruction format.

a.	`addi $t0, $t0, 0`
b.	`sw $t1, 32($t2)`

2.10.4 [5] <2.4, 2.5> For the instructions above, show the binary then hexadecimal representation of these instructions.

2.10.5 [5] <2.5> What type (I-type, R-type, J-type) instruction do the instructions above represent?

- 2.10.4 & 2.10.5
  - *(a)* addi $t1, $t1, 0 ——I-type

op	rs	rt	constant
001000	01001	01001	0
8	9	9	0

  - *(b)* sw $t1, 32($t2)——I-type

op	rs	rt	constant
101010	01010	01001	0000000000100000
2b	a	9	20

$zero: constant 0 (reg 0, also written as $0)
$at: Assembler Temporary (reg 1, or $1)
$v0, $v1: result values (reg's 2 and 3, or $2 and $3)
$a0 – $a3: arguments (reg's 4 – 7, or $4 - $7)
$t0 – $t7: temporaries (reg's 8 – 15, or $8 - $15)
$s0 – $s7: saved (reg's 16 – 23, or $16 - $23)
$t8, $t9: temporaries (reg's 24 and 25, or $24 and $25)
$k0, $k1: reserved for OS kernel (reg's 26 and 27, $26/27)
$gp: global pointer for static data (reg 28, or $28)
$sp: stack pointer (reg 29, or $29)
$fp: frame pointer (reg 30, or $30)
$ra: return address (reg 31, or $31)

16-bit immediate number or address
- rs: source or base address register
- rt: destination or source register
- Constant: -2^{15} to $+2^{15} - 1$
- Address: offset added to base address in rs

In the following problems, the data table contains various modifications that could be made to the MIPS instruction set architecture. You will investigate the impact of these changes on the instruction format of the MIPS architecture.

a.	128 registers
b.	Four times as many different instructions

2.12.1 [5] <2.5> If the instruction set of the MIPS processor is modified, the instruction format must also be changed. For each of the suggested changes above, show the size of the bit fields of an R-type format instruction. What is the total number of bits needed for each instruction?

- 2.12.1
  - *(a)*

op	rs	rt	rd	shamt	func	total
6	7	7	7	5	6	38

  - *(b)*

op	rs	rt	rd	shamt	func	total
6	5	5	5	5	8	34

In the following problems, the data table contains various modifications that could be made to the MIPS instruction set architecture. You will investigate the impact of these changes on the instruction format of the MIPS architecture.

a.	128 registers
b.	Four times as many different instructions

2.12.2 [5] <2.5> If the instruction set of the MIPS processor is modified, the instruction format must also be changed. For each of the suggested changes above, show the size of the bit fields of an I-type format instruction. What is the total number of bits needed for each instruction?

- 2.12.2
  - *(a)*

op	rs	rt	constant	total
6	7	7	16	36

  - *(b)*

op	rs	rt	constant	total
8	5	5	16	34

In the following problems, the data table contains the values for registers `$t0` and `$t1`. You will be asked to perform several MIPS logical operations on these registers.

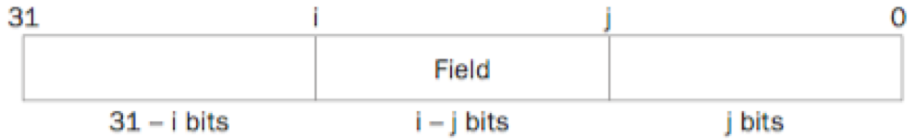
a.	`$t0 = 0xAAAAAAAA, $t1 = 0x12345678`
b.	`$t0 = 0xF00DD00D, $t1 = 0x11111111`

2.13.3 [5] <2.6> For the lines above, what is the value of `$t2` for the following sequence of instructions?

```
srl $t2, $t0, 3
andi $t2, $t2, 0xFFEF
```

- 2.12.2
  - *(a)*
    - srl: $t2=0001 0101 0101 0101 0101 0101 0101 0101 (010)
    - andi: $t2=0001 0101 0101 0101 0101 0101 0100 0101
  - *(b)*
    - srl: $t2=0001 1110 0000 0001 1011 1010 0000 0001
    - Andi $t2=0001 1110 0000 0001 1011 1010 0000 0001

The following figure shows the placement of a bit field in register $t0.



In the following problems, you will be asked to write MIPS instructions to extract the bits "Field" from register $t0 and place them into register $t1 at the location indicated in the following table.

a.

b.

2.14.1 [20] <2.6> Find the shortest sequence of MIPS instructions that extracts a field from $t0 for the constant values i = 22 and j = 5 and places the field into $t1 in the format shown in the data table.

31		22		5	4		0
15bits			17bits			5bits	

31		15	14		0
17bits			15bits		

31	30		14	13		0
1bit	17bits			14bits		

- 2.14.1
  - *(a)*
    - sll $t1, $t0, 10
    - andi $t1, $t1, 0xffff8000
  - *(b)*
    - sll $t1, $t0, 10
    - slr $t1, $t1, 1
    - ori $t1, $t1, 0x80007fff

For these problems, the table holds various binary values for register $t0. Given the value of $t0, you will be asked to evaluate the outcome of different branches.

a.	0x00101000
b.	0x80001000

2.16.4 [5] <2.7> Suppose that register $t0 contains a value from above. What is the value of $t2 after the following instructions?

```
        slt  $t2, $0,   $t0
        bne  $t2, $0,   ELSE
        j    DONE
ELSE:   addi $t2, $t2,  2
DONE:
```

$t2=(0<$t0)?1:0
If $t2!=0 jump to ELSE
Jump tp DONE
ELSE: $t2+=2

- 2.16.4
  - *(a)*
    - $t1=0x00101000>0➔$t2=1➔$t2=3
  - *(b)*
    - $t1=0x80001000<0➔$t2=0➔$t2=0

For these problems, there are several instructions that are not included in the MIPS instruction set are shown.

a.	`subi $t2, $t3, 5`	`# R[rt] = R[rs] - SignExtImm`
b.	`rpt $t2, loop`	`# if(R[rs]>0) R[rs]=R[rs]-1, PC=PC+4+BranchAddr`

2.17.3 [5] <2.7> For each instruction in the table above, find the shortest sequence of MIPS instructions that performs the same operation.

- 2.17.2
  - *(a)*
    - addi $t2, $t3, -5
  - *(b)*
    - slt $t0, $0, $t2
    - bne $t0, $0, IF
    - IF: addi $t2, $t2, -1
    - j LOOP

For these problems, the table holds MIPS assembly code fragments. You will be asked to evaluate each of the code fragments, familiarizing you with the different MIPS branch instructions.

a.	`LOOP: addi $s2, $s2, 2`
	`subi $t1, $t1, 1`
	`bne $t1, $0, LOOP`
	`DONE:`
		do{$s2+=2
		$t1+=-1}
		While($t1!=0)

b.	`LOOP: slt $t2, $0, $t1`
	`beq $t2, $0, DONE`
	`subi $t1, $t1, 1`
	`addi $s2, $s2, 2`
	`j LOOP`
	`DONE:`
		While($t1>0)
		$t1+=-1
		$s2+=2

2.17.4 [5] <2.7> For the loops written in MIPS assembly above, assume that the register `$t1` is initialized to the value 10. What is the value in register `$s2` assuming the `$s2` is initially zero?

- 2.17.4
  - *(a)*
    - $s2=20
  - *(b)*
    - $s2=20

For these problems, the table holds MIPS assembly code fragments. You will be asked to evaluate each of the code fragments, familiarizing you with the different MIPS branch instructions.

a.

```
        addi $t1, $0, 50
LOOP:   lw   $s1, 0($s0)
        add  $s2, $s2, $s1
        lw   $s1, 4($s0)
        add  $s2, $s2, $s1
        addi $s0, $s0, 8
        subi $t1, $t1, 1
        bne  $t1, $0, LOOP
```

b.

```
        addi $t1, $0, $0
LOOP:   lw   $s1, 0($s0)
        add  $s2, $s2, $s1
        addi $s0, $s0, 4
        addi $t1, $t1, 1
        slti $t2, $t1, 100
        bne  $t2, ~~$s0~~, LOOP
```

$0

2.18.4 [5] <2.7> What is the total number of MIPS instructions executed?

- 2.18.4
  - *(a)*
    - 1+7*50=351
  - *(b)*
    - 1+6*100=601

2.18.5 [5] <2.7> Translate the loops above into C. Assume that the C-level integer `i` is held in register `$t1`, `$s2` holds the C-level integer called `result`, and `$s0` holds the base address of the integer `MemArray`.

2.18.6 [5] <2.7> Rewrite the loop to reduce the number of MIPS instructions executed.

2.18.5
(a)

```
int i=50;
do {
result+=MemArray[100-2i];
result+=MemAray[101-2i];
i - -;
}while(i!=0)
```

(b)

```
int i=0;
do{
result+=MemArray[i];
i++;}while{i<100}
```

2.18.6
Consider the address directly without l

```
        addi $t1, $s0, 100
LOOP:   lw $s1, 0($s0)
        add $s2, $s2,$s1
        addi $s0, $s0, 4
        bne $t1, $s0, LOOP
```