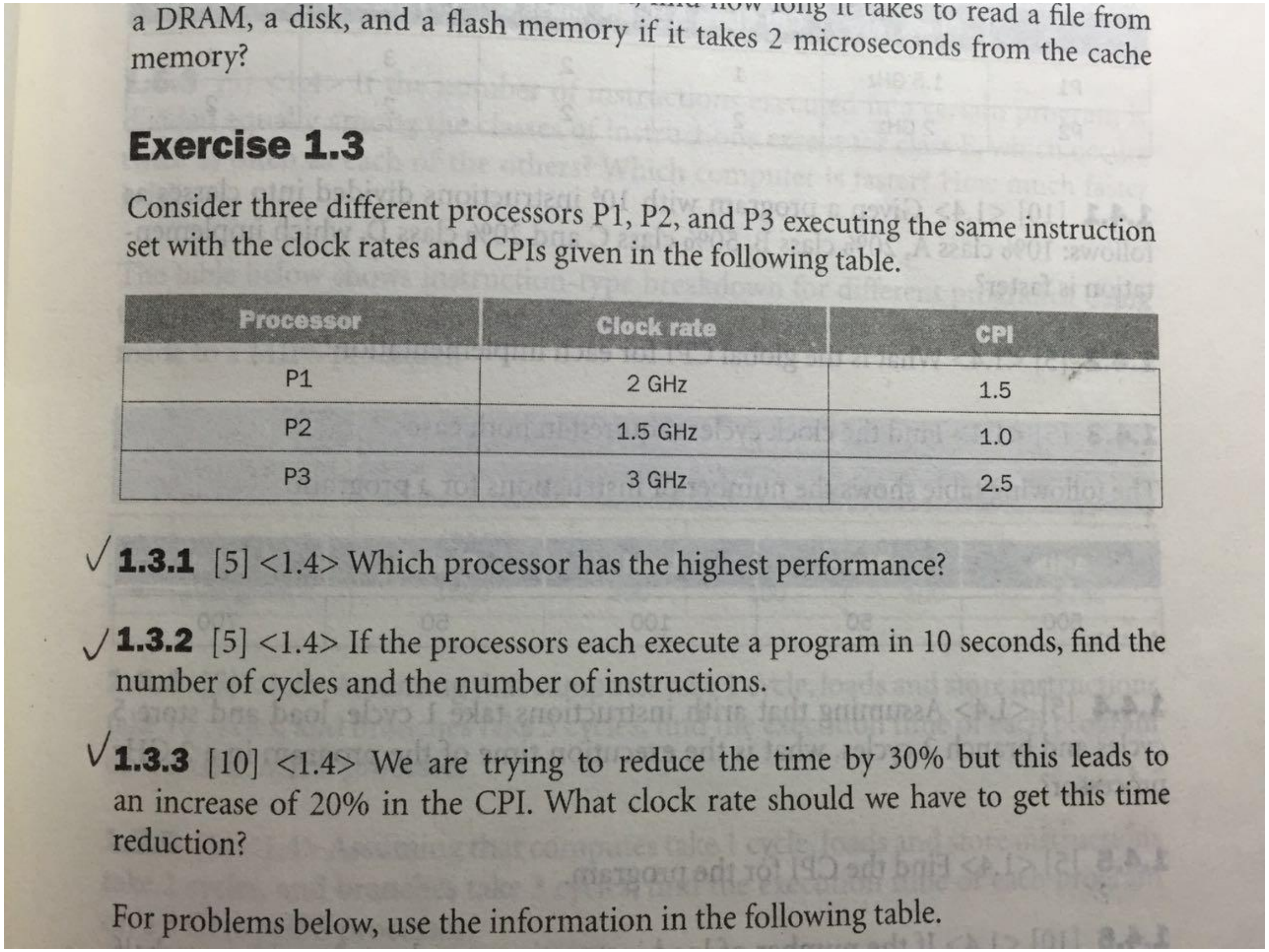
VE370 RC1

Something you need to know

- RC: Monday Noon, Thursday night, same contents
- Please do the assignments in advance
- No specific meaning of the due time on canvas
- As for projects, start early

1.3.1-1.3.3



how long it takes to read a file from a DRAM, a disk, and a flash memory if it takes 2 microseconds from the cache memory?

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
P1	2 GHz	1.5
P2	1.5 GHz	1.0
P3	3 GHz	2.5

✓ **1.3.1** [5] <1.4> Which processor has the highest performance?

✓ **1.3.2** [5] <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?

For problems below, use the information in the following table.

1.3.1-1.3.3

CPU time – for CPU Performance

- CPU execution time processing a task
  - Exclude I/O time, time spent for other tasks

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

$$\text{Clock Cycles} = \text{Instruction Count} \times \text{Clock Cycles per Instruction}(\text{CPI})$$
$$\text{CPU Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$
$$= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$

Classic CPU performance equation

1.3.1-1.3.3

- 1.3.1:

$$CPU\ Time = \frac{\#\ Inctructions\ * CPI}{Clock\ Rate}$$

- 1.3.2:

$$\#\ Cycles = Total\ time\ * Clock\ Rate$$

$$\#\ Instructions = \frac{\#\ of\ cycles}{cycle\ per\ instruction}$$

- Solutions:

	P1	P2	P3
CPU Time ($\times I \times 10^{-9} s$)	0.75	0.66	0.83
# Cycles ($\times 10^{-10}$)	2	1.5	3
# Instructions ($\times 10^{10}$)	1.33	1.5	1.2

1.3.1-1.3.3

- 1.3.3:

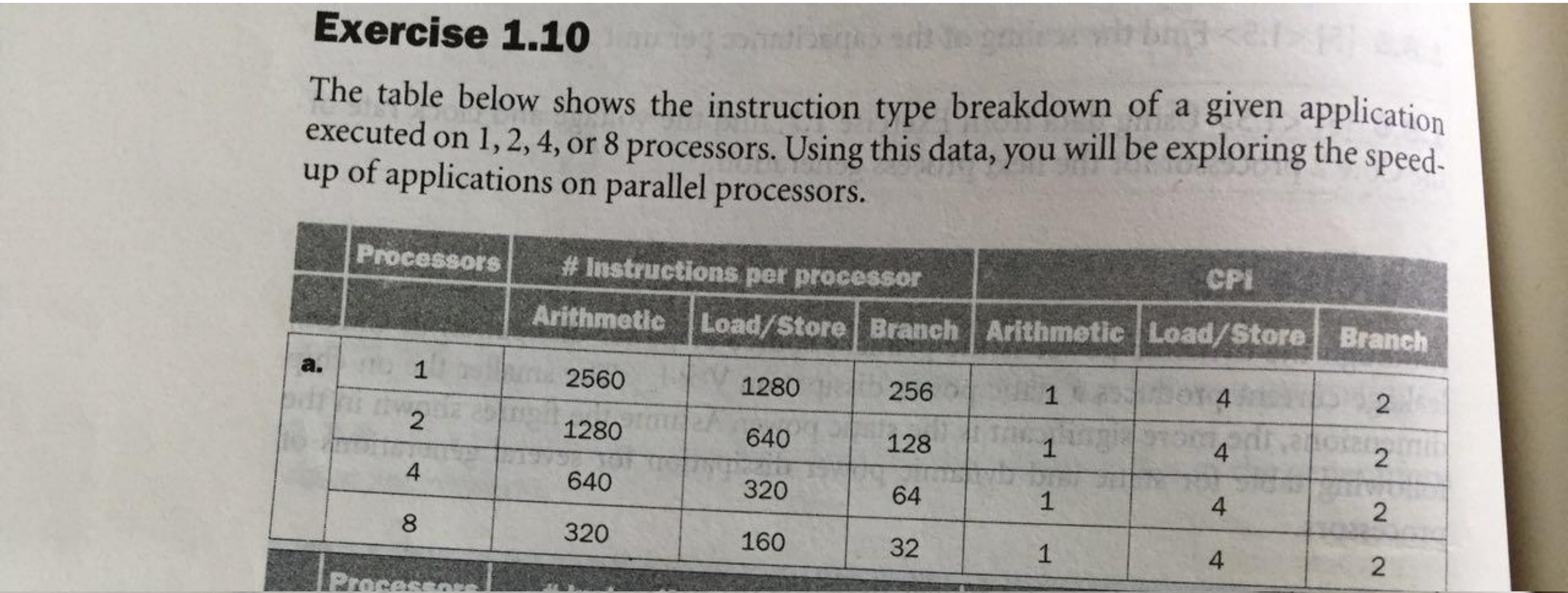
$$Clock\ Rate = \frac{\#\ Inctructions\ \ *CPI}{CPU\ Time}$$

- Solutions:

$$Clock\ Rate' = \frac{\#\ Inctructions\ *CPI'}{CPU\ Time'} = \frac{\#\ Inctructions\ *1.2*CPI}{0.7*CPU\ Time} = 1.71\ \text{Clock Rate}$$

So we need to increase the clock rate by 71%

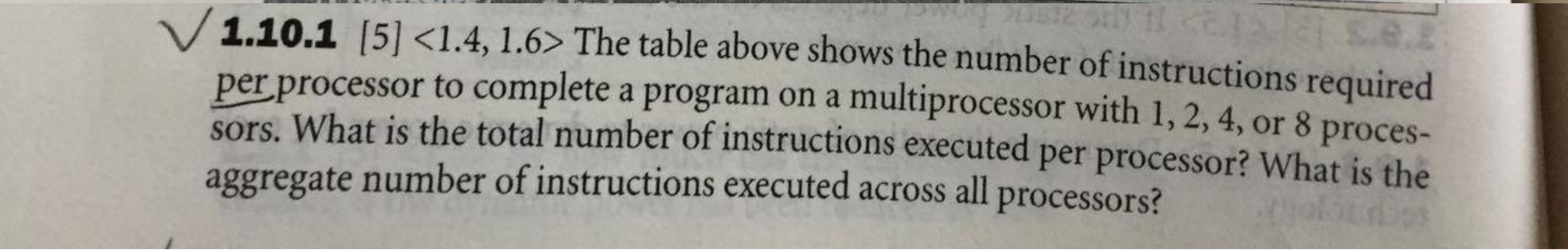
1.10.1(a)



Exercise 1.10

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	# 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	4	2
	4	640	320	64	1	4	2
	8	320	160	32	1	4	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?

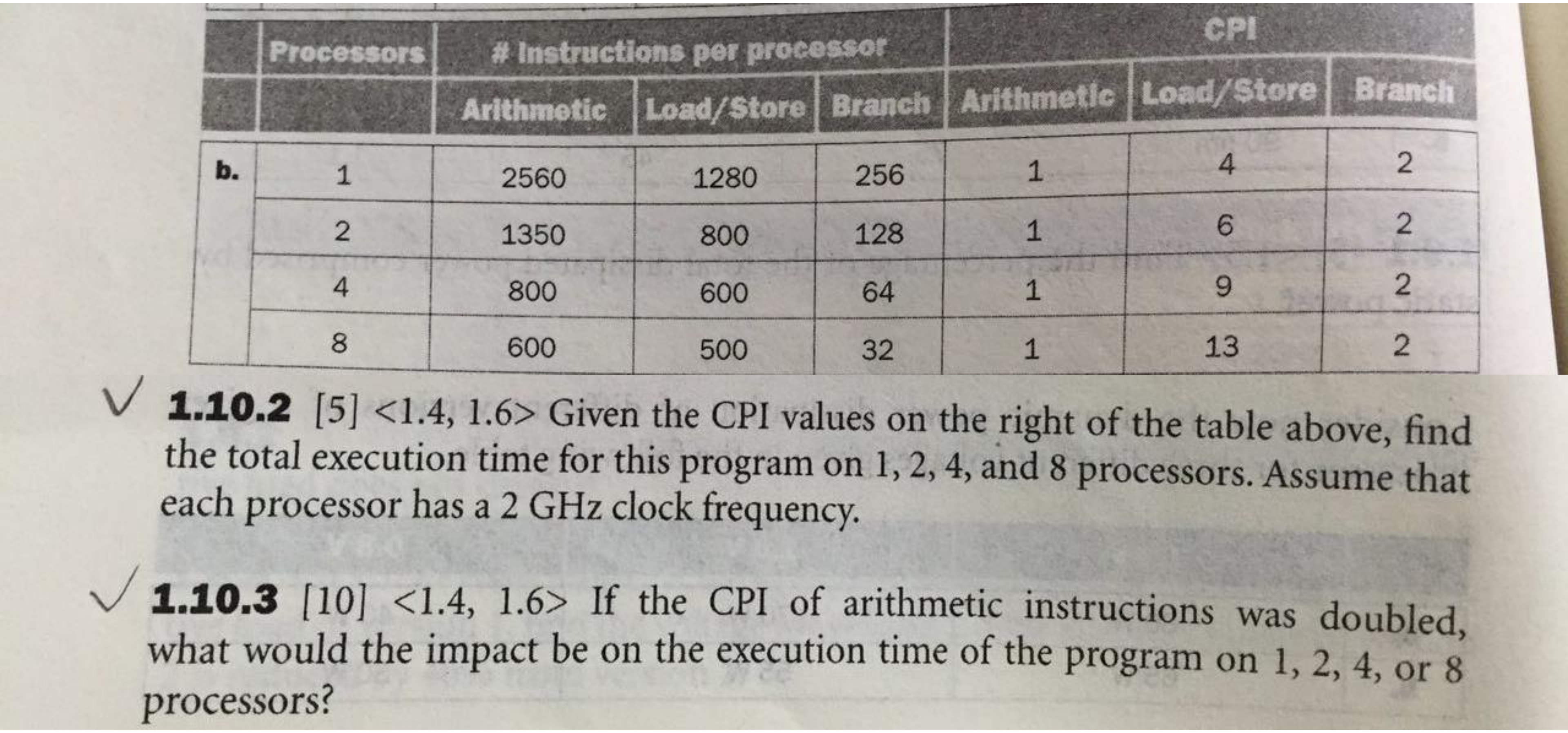
1.10.1(a)

- #instructions = #ins_Arithmetic + #ins_Load/Save + #ins_branch
- Take the 4 processor as an example:

Instruction per processor = 640 + 320 + 64 = 1024

Total Instructions: Ins/Processor * num of Processor = 1024 * 4 = 4096

1.10.2-1.10.3(a)



	Processors	# Instructions per processor			CPI		
		Arithmetic	Load/Store	Branch	Arithmetic	Load/Store	Branch
b.	1	2560	1280	256	1	4	2
	2	1350	800	128	1	6	2
	4	800	600	64	1	9	2
	8	600	500	32	1	13	2

✓ **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 arithmetic instructions was doubled, what would the impact be on the execution time of the program on 1, 2, 4, or 8 processors?

1.10.2-1.10.3(a)

- Again, take the 4 processor as an example:
- 1.10.2(a):

$$CPU\ Time = \frac{\#\ Inctructions\ \ * CPI}{Clock\ Rate}$$

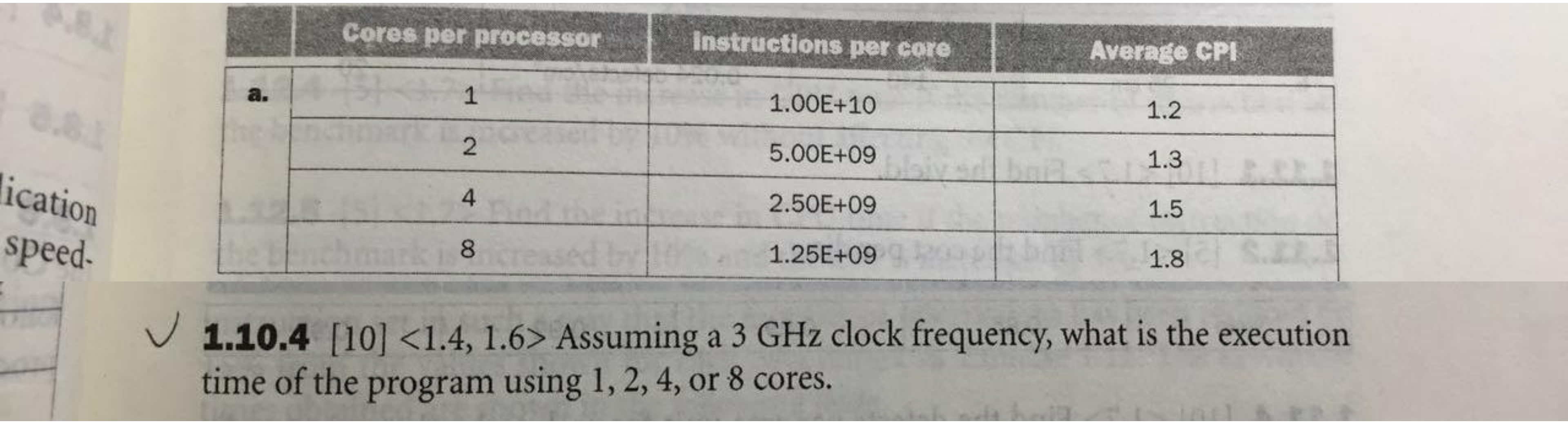
$$CPU\ Time = \frac{640*1+320*4+64*2}{2GHz} = 1.024*10^{-6}s$$

- 1.10.3(a):

$$CPU\ Time = \frac{640*2+320*4+64*2}{2GHz} = 1.344*10^{-6}s$$

NOTE: Do not multiple the # of processor!

1.10.4(a)



	Cores per processor	Instructions per core	Average CPI
a.	1	1.00E+10	1.2
	2	5.00E+09	1.3
	4	2.50E+09	1.5
	8	1.25E+09	1.8

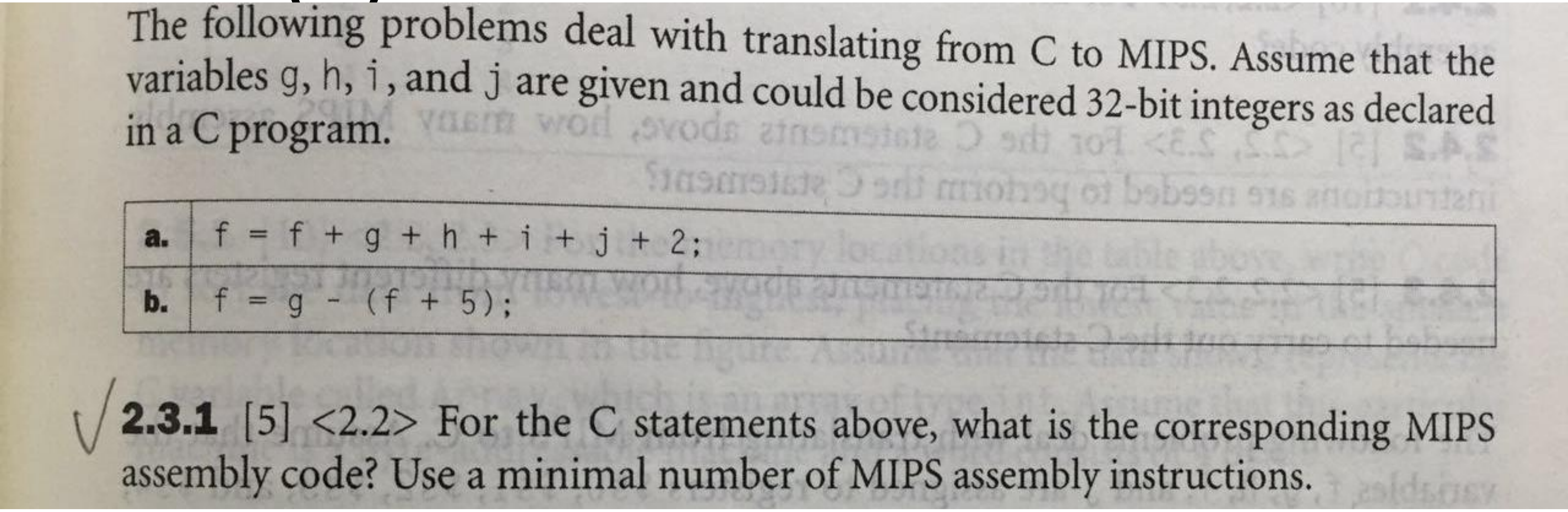
✓ **1.10.4** [10] <1.4, 1.6> Assuming a 3 GHz clock frequency, what is the execution time of the program using 1, 2, 4, or 8 cores.

- 1.10.4(a):

$$CPU\ Time = \frac{\#\ Inctructions\ *CPI}{Clock\ Rate}$$

	1	2	4	**8**
CPU Time (s)	4	2.17	1.25	0.75

2.3.1(b)



The following problems deal with translating from C to MIPS. Assume that the variables g, h, i, and j are given and could be considered 32-bit integers as declared in a C program.

a.	`f = f + g + h + i + j + 2;`
b.	`f = g - (f + 5);`

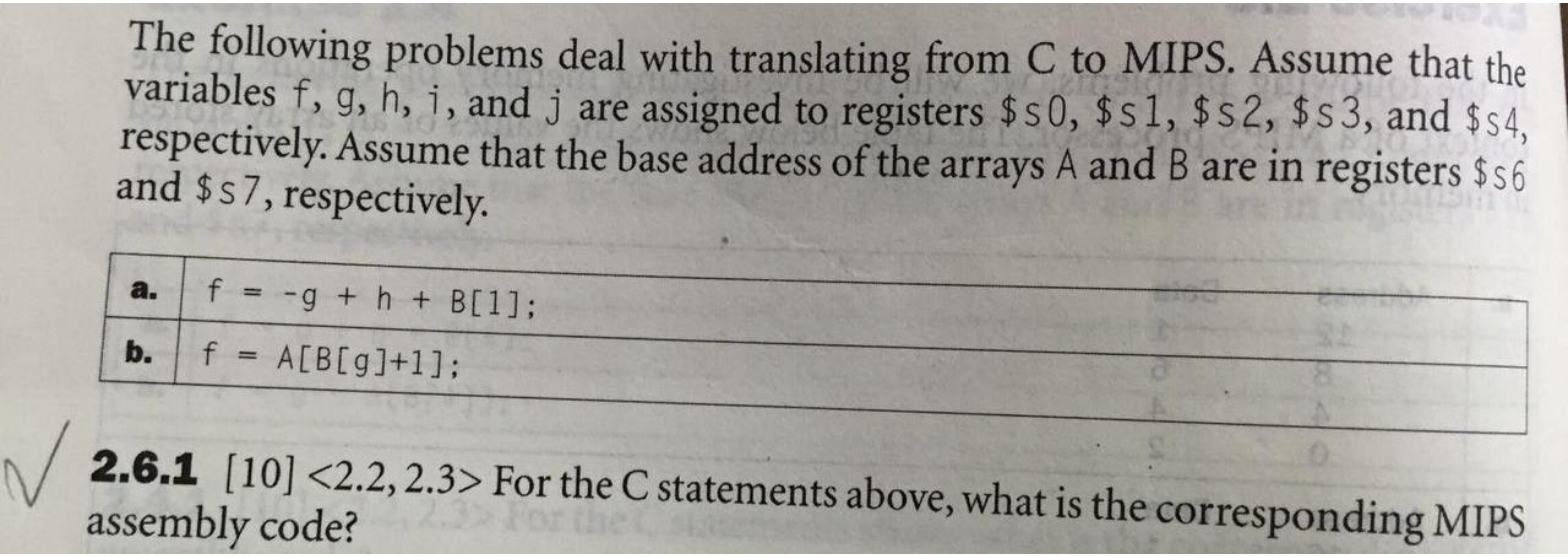
√ **2.3.1** [5] <2.2> For the C statements above, what is the corresponding MIPS assembly code? Use a minimal number of MIPS assembly instructions.

- 2.3.1(b):

Assume f as $s0, and g as $s1:

```
addi $t0, $s0, 5          temp1 = f+5;
sub $s0, $s1, $t0         f = g – temp1;
```

2.6.1(b)



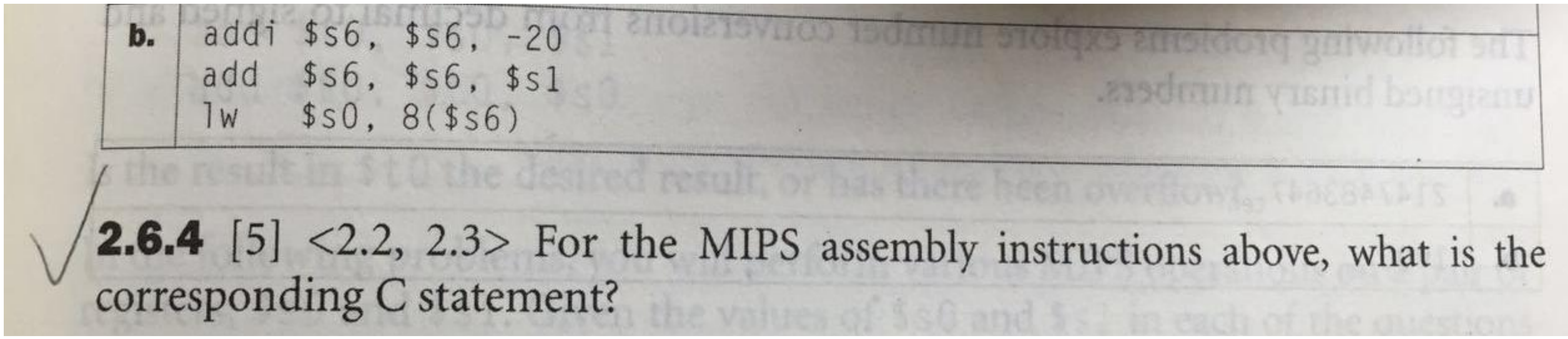
The following problems deal with translating from C to MIPS. Assume that the variables f, g, h, i, and j are assigned to registers $s0, $s1, $s2, $s3, and $s4, respectively. Assume that the base address of the arrays A and B are in registers $s6 and $s7, respectively.

a.	`f = -g + h + B[1];`
b.	`f = A[B[g]+1];`

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

sll $t0, $s1, 2	tmp1 = g * 4;	calculate the offset for B[g]
add $t0, $t0, $s7	tmp1 += B;	Get address for B[g]
lw $t1, 0($t0)	tmp2 = B[g];	load the content of B[g]
addi $t1, $t1, 1	tmp2 ++;	content of B[g] +1
sll $t1, $t1, 2	tmp2 *= 4;	Calculate the offset
add $t1, $t1, $s6	tmp2 += A;	get the address of A[B[g]+1]
lw $s0, 0($t1)	f = A[tmp2];	load the content to f

2.6.4(b)



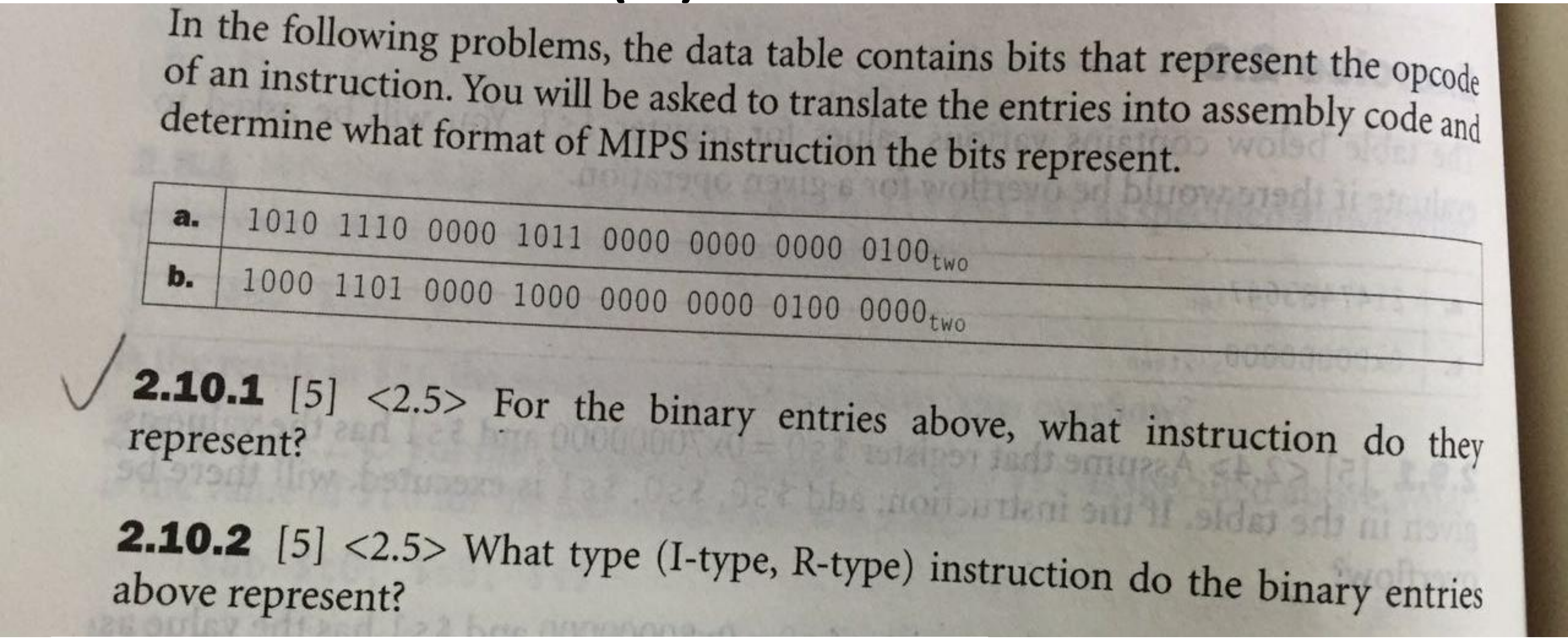
```
b.  addi $s6, $s6, -20
    add  $s6, $s6, $s1
    lw   $s0, 8($s6)
```

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

addi \$s6, \$s6, -20	s6=s6-20;	Address of A minux 20: Address of A[-5]
add \$s6, \$s6, \$s1	s6=s6+s1;	Then Address of A +g : Address of A[-5+g/4]
lw \$s0, 8(\$s6)	s0=s6[2];	load the content of A[-5+g/4+2] to f

f=A[-5+g/4+2]=A[g/4-3]

2.10.1-2.10.2(a)



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

a.	1010 1110 0000 1011 0000 0000 0000 0100_{two}
b.	1000 1101 0000 1000 0000 0000 0100 0000_{two}

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

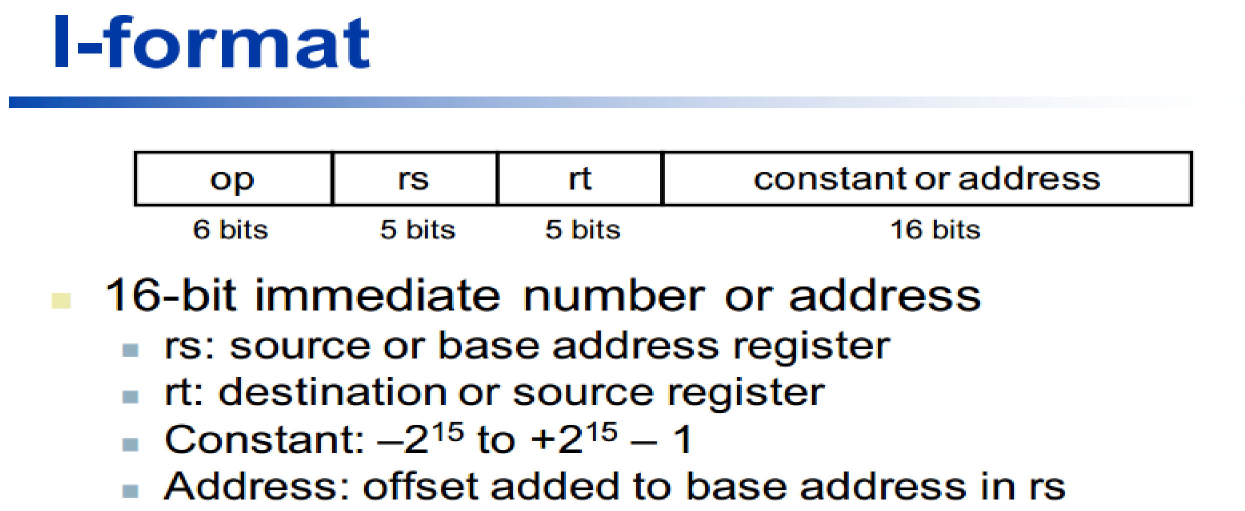
2.10.2 [5] <2.5> What type (I-type, R-type) instruction do the binary entries above represent?

op	rs	rt	constant or address
6 bits	5 bits	5 bits	16 bits

I - Type

1010 1110 0000 1011 0000 0000 0000 0100
101011 10000 01011 0000000000000100
2b 16 11 4

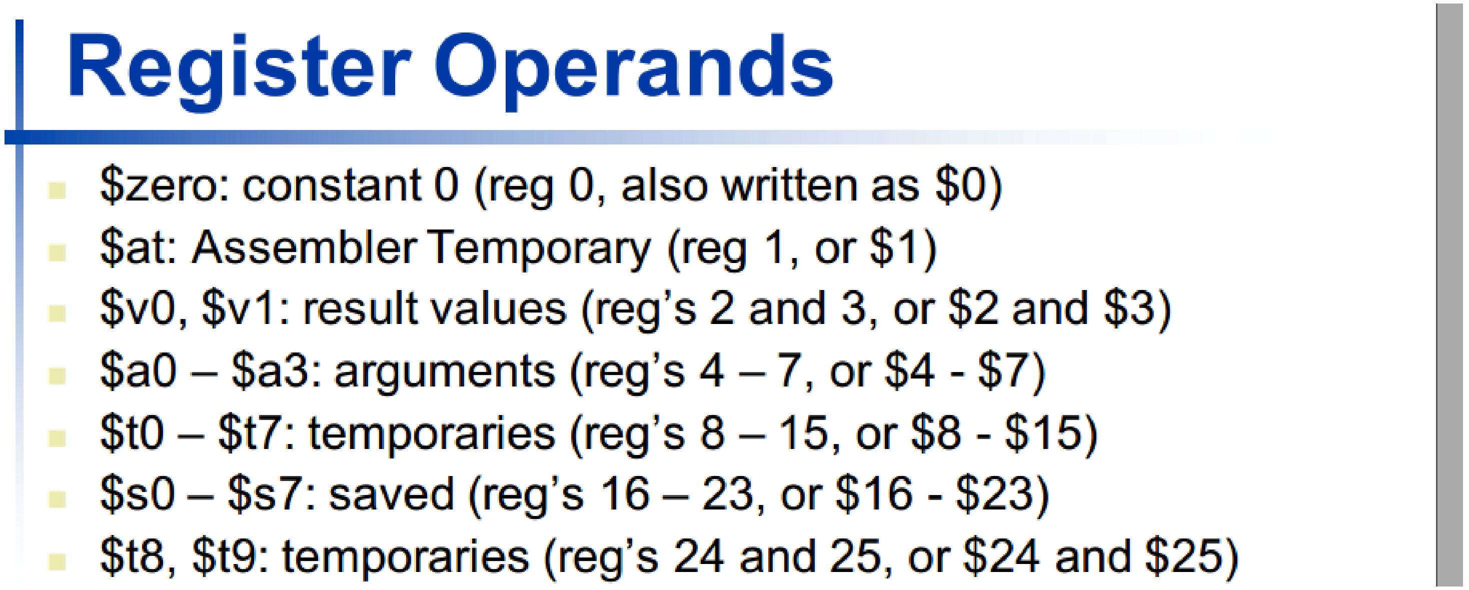
2.10.1-2.10.2(a)



I-format

op	rs	rt	constant or address
6 bits	5 bits	5 bits	16 bits

- 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



Register Operands

- $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)

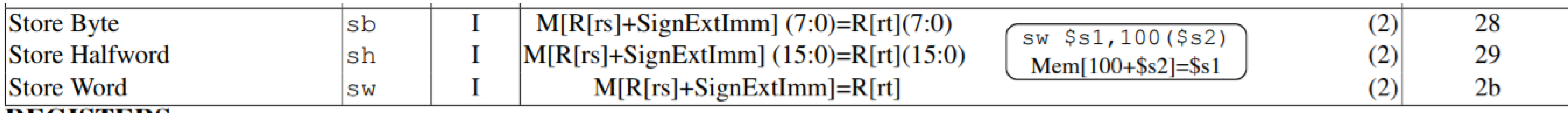
2.10.1-2.10.2(a)

op	rs	rt	constant or address
6 bits	5 bits	5 bits	16 bits

I - Type

```
1010 1110 0000 1011 0000 0000 0000 0100
101011 10000 01011  0000000000000100
2b          16        11        4
sw          $s0      $t3      4
```

Pay attention to the MIPS Reference Card!

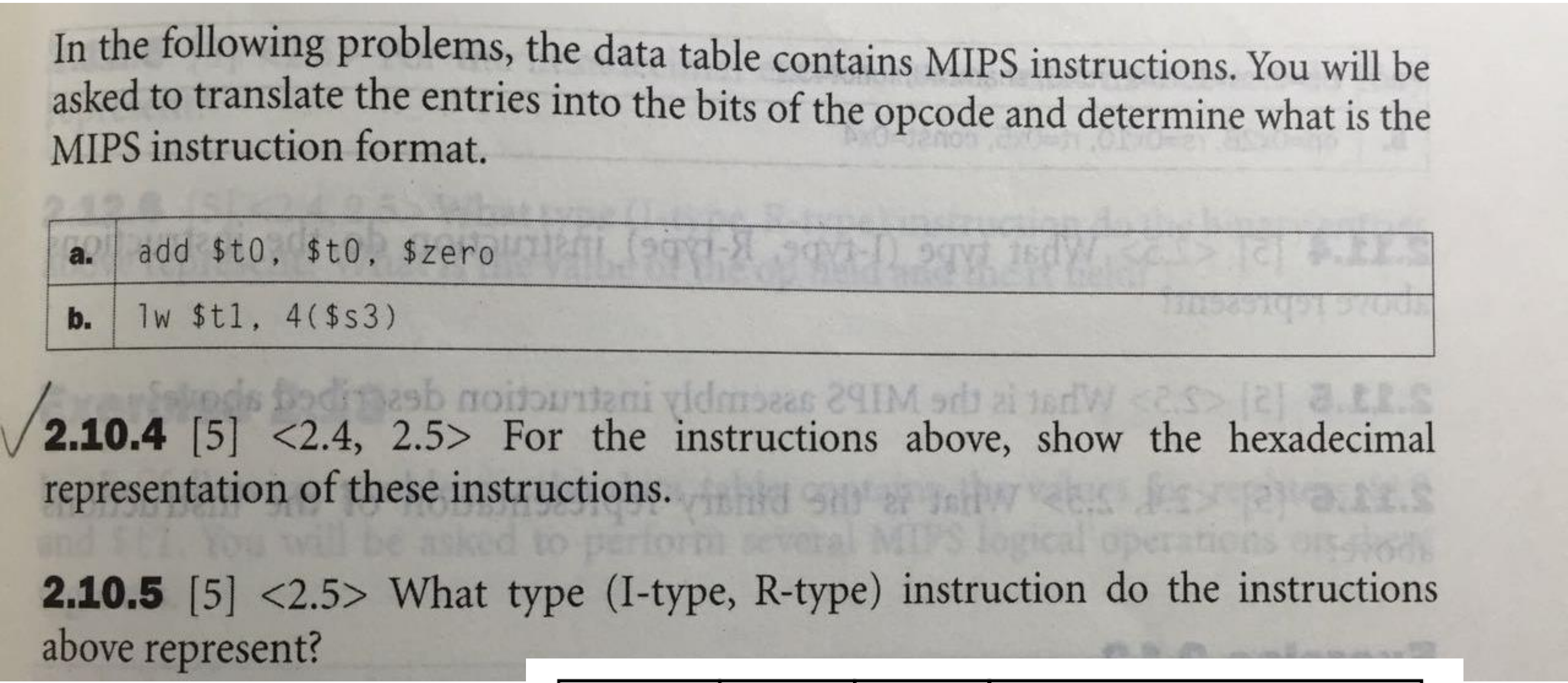


Store Byte	sb	I	M[R[rs]+SignExtImm] (7:0)=R[rt](7:0)		(2)	28
Store Halfword	sh	I	M[R[rs]+SignExtImm] (15:0)=R[rt](15:0)	sw $s1,100($s2) Mem[100+$s2]=$s1	(2)	29
Store Word	sw	I	M[R[rs]+SignExtImm]=R[rt]		(2)	2b

Solution: sw $t3, 4($s0)

Note: Do not divide by 4 here

2.10.4-2.10.5(b)

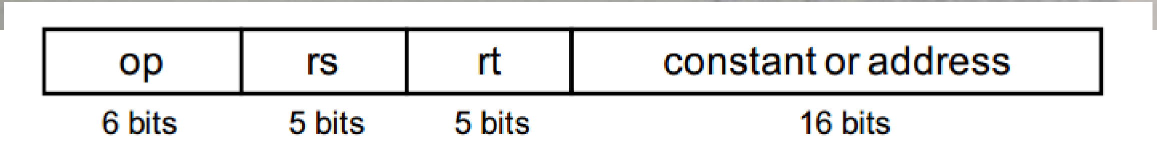


In the following problems, the data table contains MIPS instructions. You will be asked to translate the entries into the bits of the opcode and determine what is the MIPS instruction format.

a.	`add $t0, $t0, $zero`
b.	`lw $t1, 4($s3)`

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

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

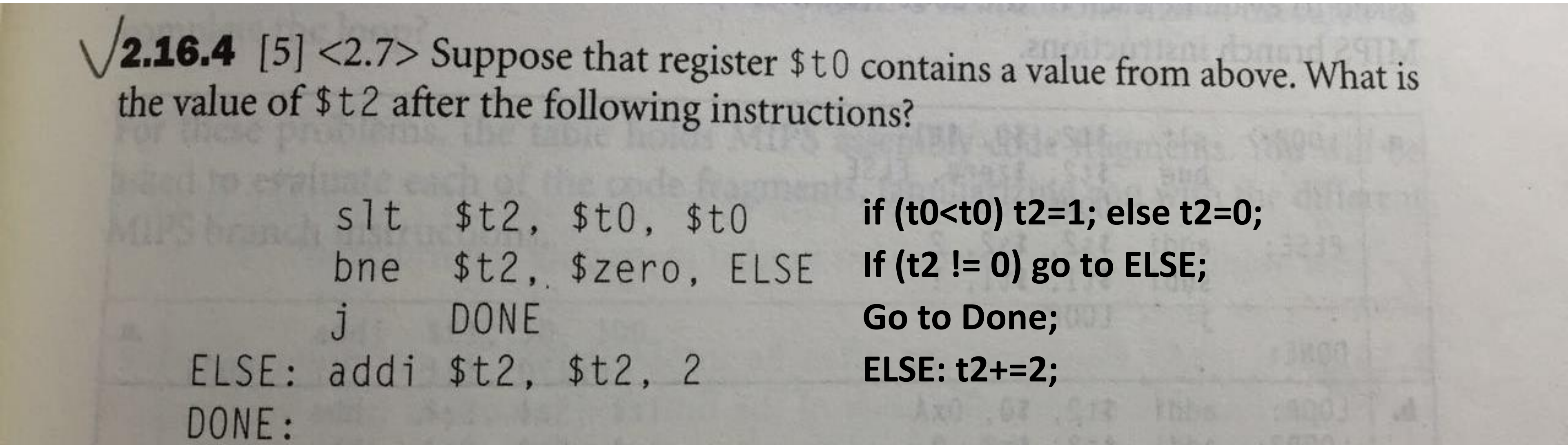


op	rs	rt	constant or address
6 bits	5 bits	5 bits	16 bits

Solution: I Type

100011 10011 01001 0000000000001000
1000 1110 0110 1001 0000 0000 0000 1000
8 E 6 9 0 0 0 4

2.16.4 (b)



√**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, $t0, $t0
        bne  $t2, $zero, ELSE
        j    DONE
ELSE: addi $t2, $t2, 2
DONE:
```

if (t0<t0) t2=1; else t2=0;

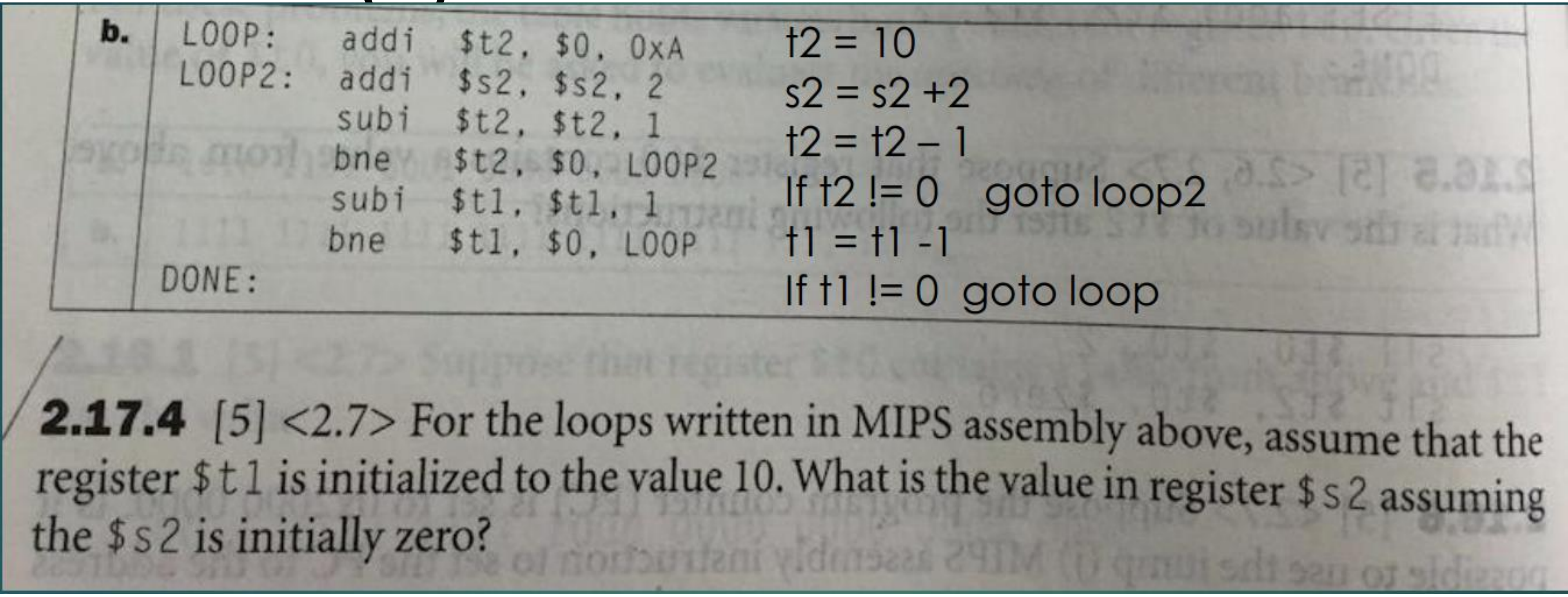
If (t2 != 0) go to ELSE;

Go to Done;

ELSE: t2+=2;

Solution: $t2 = 0

2.17.4 (b)



```
b.  LOOP:   addi  $t2, $0, 0xA       t2 = 10
    LOOP2:  addi  $s2, $s2, 2        s2 = s2 +2
            subi  $t2, $t2, 1        t2 = t2 – 1
            bne   $t2, $0, LOOP2     If t2 != 0    goto loop2
            subi  $t1, $t1, 1        t1 = t1 -1
            bne   $t1, $0, LOOP      If t1 != 0  goto loop
    DONE:
```

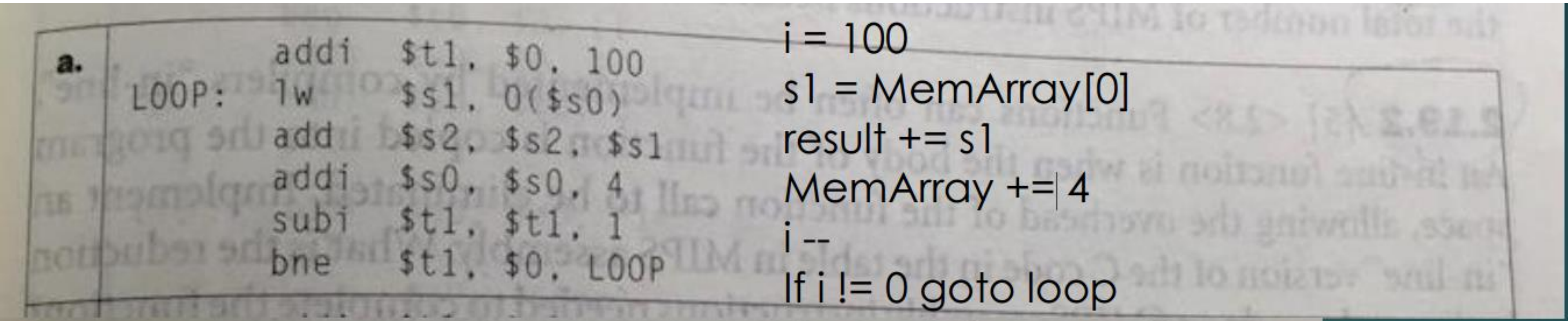
2.17.4 [5] <2.7> For the loops written in MIPS assembly above, assume that the register `$t1` is initialized to the value 10. What is the value in register `$s2` assuming the `$s2` is initially zero?

```
do{
        t2 = 10;
        do{
                s2 += 2;
                t2 --;
        } while (t2 != 0)
        t1 --;
} while (t1 != 0)
```

2 * 10 *10 =200

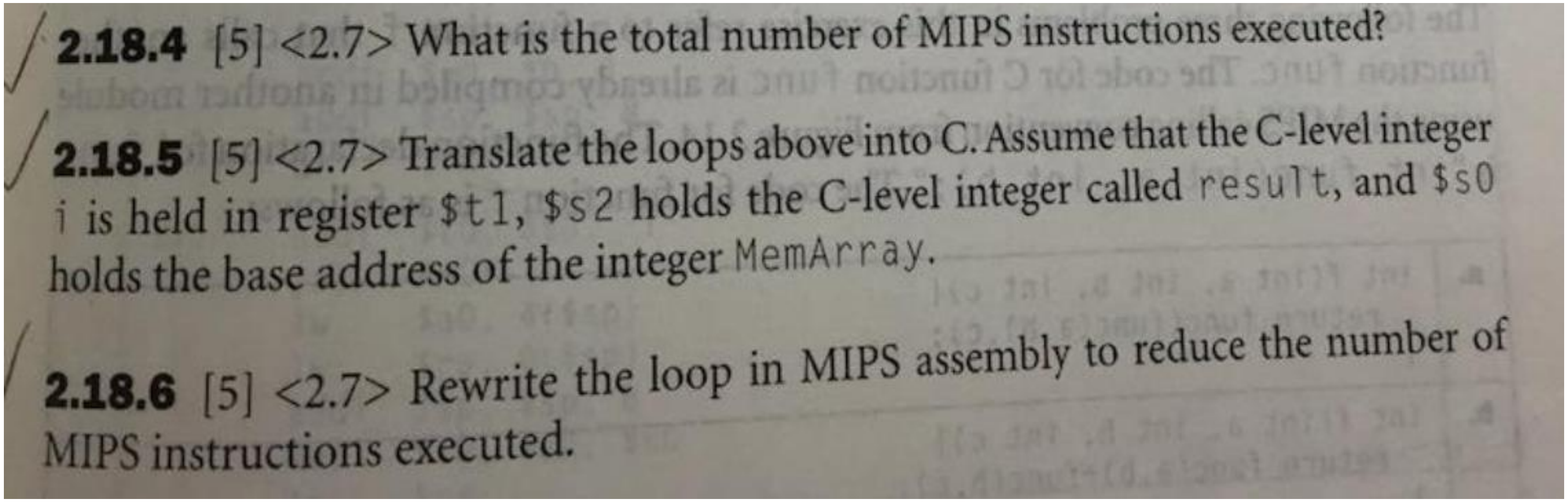
$s2 = 200

2.18.4-2.18.6 (a)



```
a.          addi   $t1, $0, 100
    LOOP:   lw     $s1, 0($s0)
            add    $s2, $s2, $s1
            addi   $s0, $s0, 4
            subi   $t1, $t1, 1
            bne    $t1, $0, LOOP
```

```
i = 100
s1 = MemArray[0]
result += s1
MemArray += 4
i --
If i != 0 goto loop
```

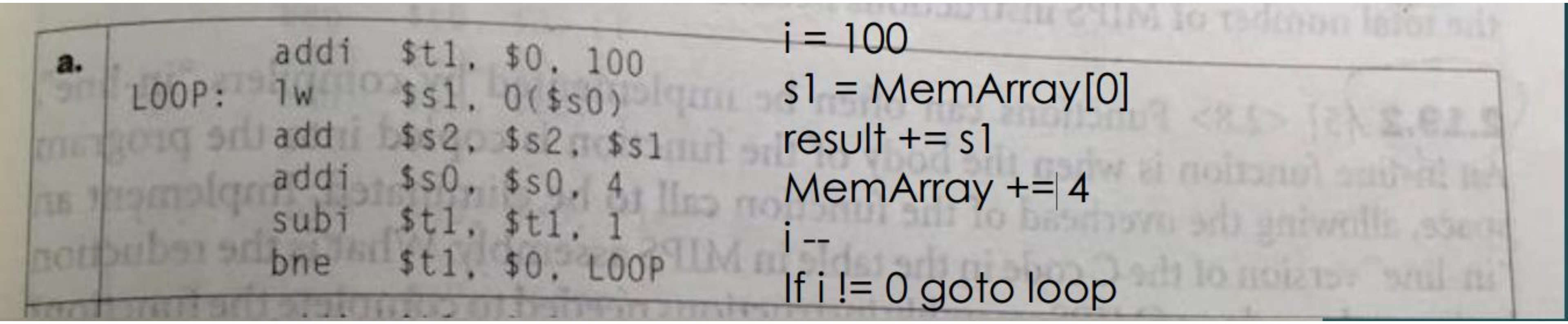


2.18.4 [5] <2.7> What is the total number of MIPS instructions executed?

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 in MIPS assembly to reduce the number of MIPS instructions executed.

2.18.4-2.18.6 (a)



```
a.          addi   $t1, $0, 100
     LOOP:  lw     $s1, 0($s0)
            add    $s2, $s2, $s1
            addi   $s0, $s0, 4
            subi   $t1, $t1, 1
            bne    $t1, $0, LOOP
```

i = 100
s1 = MemArray[0]
result += s1
MemArray += 4
i --
If i != 0 goto loop

2.18.4 (b)
5*100+1=501

2.18.5 (b)

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

2.18.6 (b)

```
            addi $t1, $s0, 400
Loop:        lw $s1, 0($s0)
            add $s2, $s2, $s1
            addi $s0, $s0, 4
            bne $t1, $s0, Loop
```

(Leave the counter i, and directly consider the address)