Exercise 1.3

1.3.1 The overall performance can be assessed by IPS (instructions per second)

For P_1: $IPS_{P_1} = \frac{\text{Clock Rate}}{CPI} = \frac{2\times10^9}{1.5} \approx 1.33\times10^9$

For P_2: $IPS_{P_2} = \frac{1.5\times10^9}{1.0} = 1.5\times10^9$

For P_3: $IPS_{P_3} = \frac{3\times10^9}{2.5} = 1.2\times10^9$

So, Processor P_2 has the highest performance.

1.3.2. Total cycles = Clock Rate x Time

Total instructions = Total cycles / CPI

For P_1: Total cycles $= 2\times10^9\times1 = 2\times10^{10}$

Total instructions $= 2\times10^{10} / 1.5 = 1.33\times10^9$

For P_2: Total cycles $= 1.5\times10^9\times1 = 1.5\times10^{10}$

Total instructions $= 1.5\times10^{10} / 1.0 = 1.5\times10^{10}$

For P_3: Total cycles $= 3\times10^9\times1 = 3\times10^{10}$

Total instructions $= 3\times10^{10} / 2.5 = 1.2\times10^{10}$

1.3.3. Assume the original time has 1 unit

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

Corresponding Modify:

$$\text{New Clock Rate} = \text{Original Clock Rate} \times \frac{1.2}{0.7}$$

For P1: $NCR = 2\times10^9 \times \frac{1.2}{0.7} = 3.43\times10^9$

For P2: $NCR = 1.5\times10^9 \times \frac{1.2}{0.7} = 2.57\times10^9$

For P3: $NCR = 3\times10^9 \times \frac{1.2}{0.7} = 5.14\times10^9$

1.3.4. $$IPC = \frac{\text{Number of Instructions}}{\text{Total Cycles}}$$

For P1: $IPC = \frac{2\times10^9}{2\times10^9\times7} = 1.43$

For P2: $IPC = \frac{3\times10^9}{1.5\times10^9\times10} = 2$

For P3: $IPC = \frac{9\times10^9}{3\times10^9\times9} = 3.33$

1.3.5.

$$\text{Clock Rate} = \frac{3\times10^9\times\frac{1}{2}}{7} = 2.14\times10^9$$

1.3.6:

$$\text{No. Instructions} = \frac{1.5 \times 10^9 \times 9}{\frac{1}{2}} = 27 \times 10^9$$

Exercise 1.4

1.4.1

$$\text{Execution Time} = \frac{\text{No. Instructions} \times \text{Average CPI}}{\text{Clock Rate}}$$

For P1:

$$\text{Average CPI} = 0.1 \times 1 + 0.2 \times 2 + 0.5 \times 3 + 0.2 \times 4 = 2.8$$

$$\text{Execution Time} = \frac{10^6 \times 2.8}{1.5 \times 10^9} = 1.87 \times 10^{-3} s$$

For P2:

$$\text{Average CPI} = 0.1 \times 2 + 0.2 \times 2 + 0.5 \times 2 + 0.2 \times 2 = 2$$

$$\text{Execution Time} = \frac{10^6 \times 2}{2 \times 10^9} = 1.0 \times 10^{-3} s$$

Conclusion: Processor P2 is faster.

1.4.2

For P1: Global CPI = 2.8

For P2: Global CPI = 2

1.4.3 Clock Cycles = No. Instructions x Average CPI

For P1: $10^6 \times 2.8 = 2.8 \times 10^6$

For P2: $10^6 \times 2.0 = 2.0 \times 10^6$

1.4.4.

Total Cycles = $500 \times 1 + 150 \times 5 + 50 \times 2 = 1350$

Execution Time = $\frac{1350}{2 \times 10^9} = 6.75 \times 10^{-7}$

1.4.5.

$CPI = \frac{1350}{700} = 1.93$

1.4.6.

Speed-up = $\frac{\text{old Total Cycles}}{\text{New Total Cycles}} = \frac{1350}{1100} = 1.23$

Exercise 2.4

2.4.1

For a.

```
lw   $t0, 16($s7);
add  $t1, $s1, $s2;
add  $s0, $t1, $t0;
```

For b.:

```
lw   $t0, 16($s7);
sll  $t0, $t0, 2;
add  $t0, $t0, $s6;
lw   $t1, 0($t0);
sub  $s0, $s1, $t1;
```

2.4.2.

For a. 3 MIPS Instructions (One lw + Two add)

For. b. 5 MIPS Instructions (Two lw + one sll + One add + one sub)

2.4.3.

For a. 6 different registers:

$s0, $s1, $s2, $s7, $t0, $t1

For b. 6 different registers:

$s0, $s1, $s6, $s7, $t0, $t1

2.4.4.

For a. f = f + g + h + i + j;

For b. f = A[1]

2.4.5.

To be truth, these 2 assembly codes are already the optimal and the instructions are minimum.

2.4.6.

Original Code Utilization:

For a - 5 registers used

$s0, $s1, $s2, $s3, $4

For b - 2 registers used

$s0, $s6

These 2 are already optimized.

Exercise 2.11

2.11.1-

For a.

1010 1110 0000 1011 1111 1111 1111 1100

For b.

1000 1101 0000 1000 1111 1111 1100 0000

2.11.2

For a. 2920022012

For b. 2366177216

2.11.3.

For a: sw $11 0xFFFC($16)

For b: lw $8 0xFFC0($8)

2.11.4.

For a: R-type

For b: I-type

2.11.5

For a: add $s3, $s1, $t2

For b: sw $t5, 4($s16)

2.11.6.

For a. 000000 0001 0010 0011 00000 100000

For b. 101011 10000 00101

0000 0000 0000 0100

Exercise 2.13

2.13.1

For a - 0x5775S778

For b. 0xFEFF-FEDE

2.13.2

For a, 0x5555555o

For b. 0xEADFEEDO

2.13.3.

For a. 0xAAAA

For b. 0xBFCD

2.13.4.

For a. 0x0001SB5A

For b. 0x0000oDo

2.13.5

For a. 0XEFEF0000

for b. 0x0000

2.13.6.

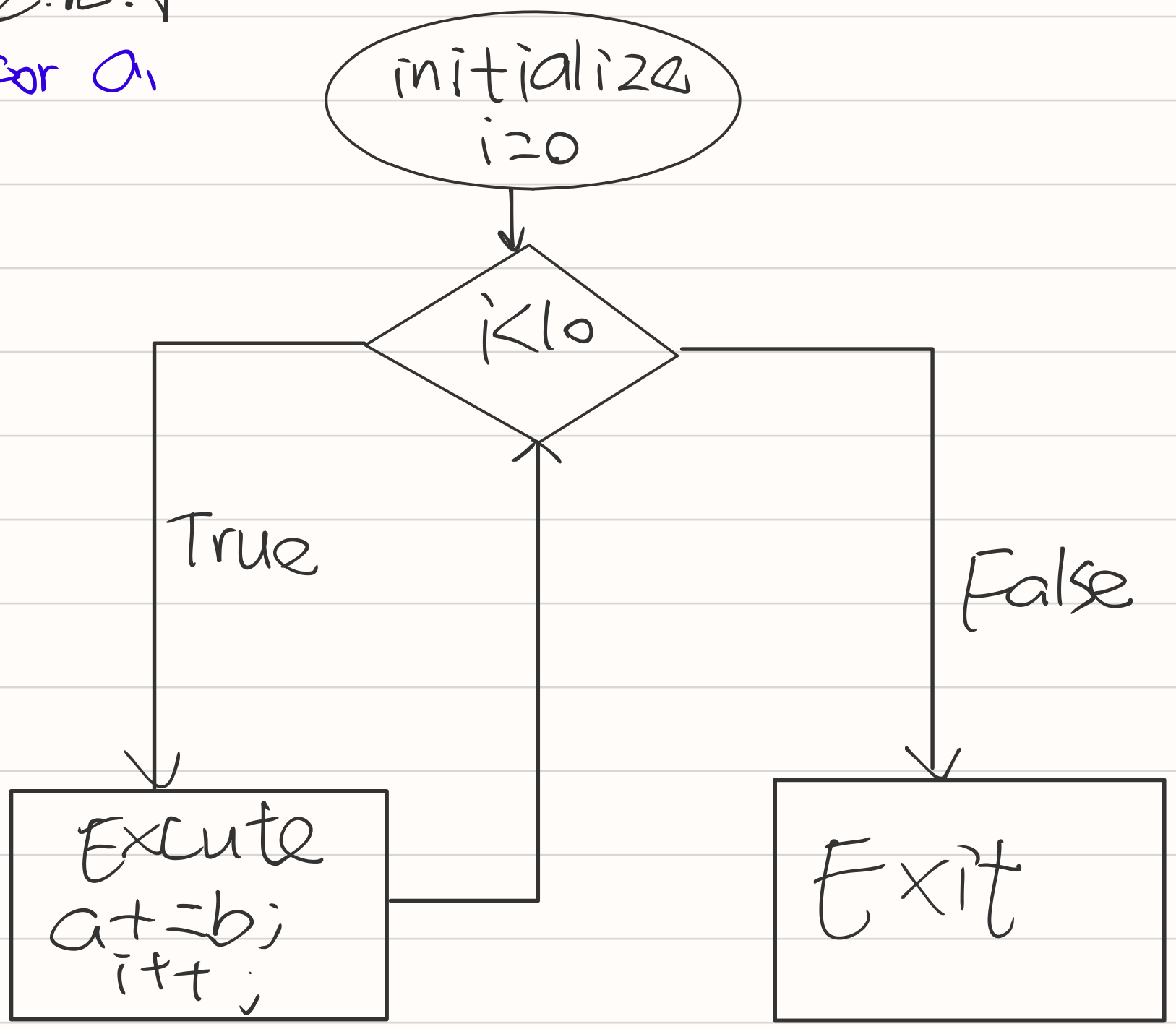
For a. 0XEFEFFFFF

For b. 0x00F0

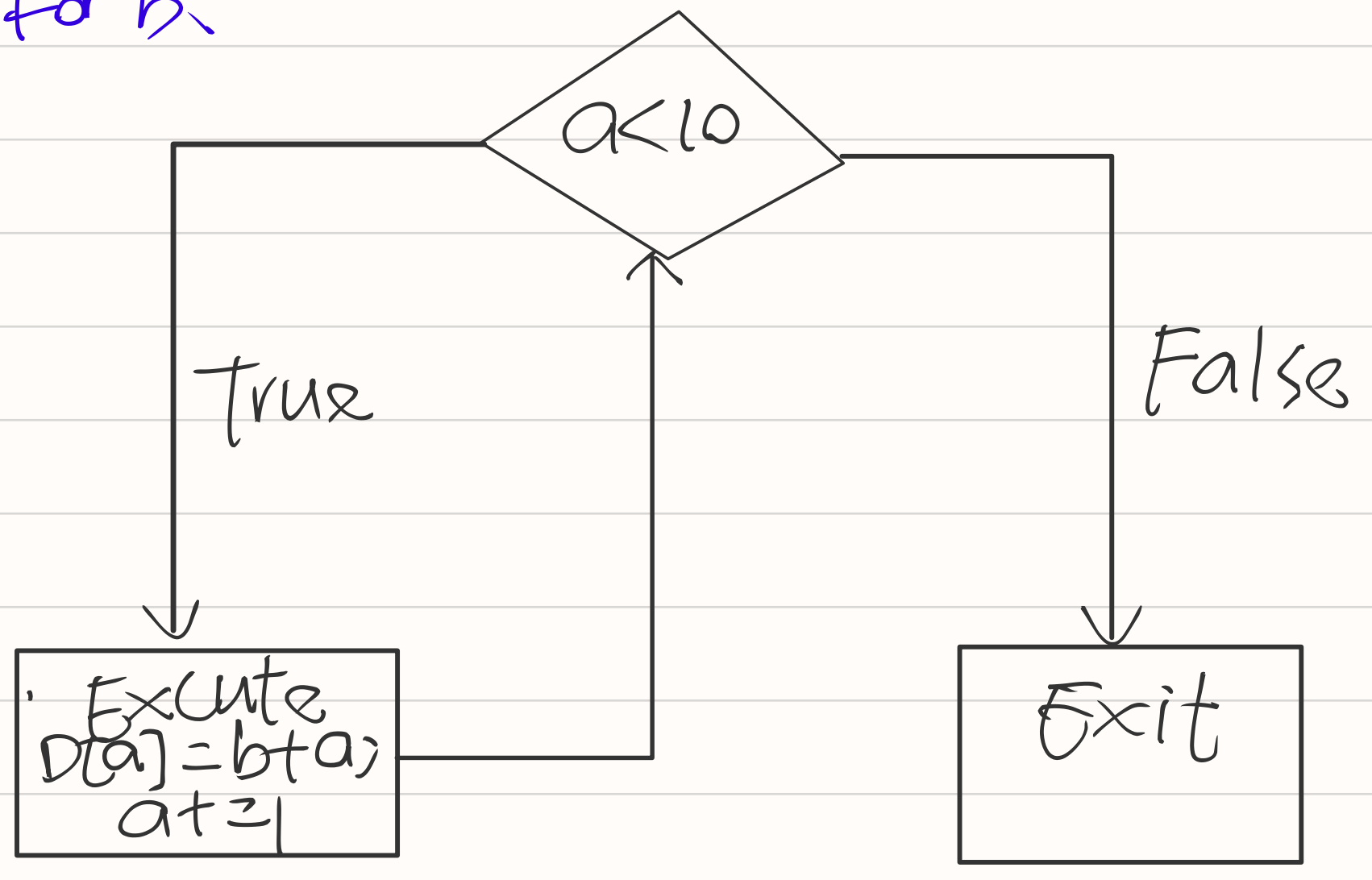
Exercise 2.1B

2.B.1

For a.



For b.



2.18.2

For a, li $t0, 0

```
for_loop:
    bge $t0, 10, exit_for
    add $s0, $s0, $s1
    addi $t0, $t0, 1
     j for_loop
exit_for
```

For b.

```
while_loop:
    bge $s0, 10, exit_while
    add $t1, $s1, $s0
     sw $t1, 0($s2)
    addi $s0, $s0, 1
    addi $s2, $s2, 4
     j while_loop
exit_while:
```

2.18.3.

For a. 1+ # of loop x4

For b. # of loop x6

After init a. b and D

. For a. 1+4x10=41

For b. 1

2.184.

For a. 501

For b. 301

2.185.

For a.
```
int i, result=0;
int *MemArray = ...;
for(i=0; i<100; i++)
  { result += MemArray[i]; }
```

For b.
```
int i, result=0;
int *MemArray = ...;
for(i=0; i<50; i++)
 { result += MemArray[2i] + MemArray[2i+1]; }
```

2.186.

For a.
```
addi $t1, $s0, 100
addi $s0, $s0, 400
Loop:
  subi $s0, $s0, 4
  lw   $s1, 0($s0)
  add  $s2, $s2, $s1
  bne  $s0, $t1, Loop.
```

For b.

```
addi $t1, $s0, 400
Loop:
 lw $s1, 0($s0)
 lw $t2, 4($s0)
 add $s2, $s2, $s1
 add $s2, $s2, $t1
 addi $s0, $s0, 8
 bne $s0, $t1, Loop.
```

Exercise 2.27

2.27.1.

For a. add $s0, $s1, $s2

For b. beq $t0, $t1, label

The address of Label is calculated relative to the current PC

2.27.2.

For a. R-type

For b. I-type

2.27.3.

Benefits: Reduced code size
Efficiency in Branching
Position-Independent Code

Drawbacks: Limited Range
Potential for Errors

```
Loop_start:
    lw $t0, 0($s0)
  addi $s0, $s0, 4
   bne $t0, $zero, Loop_start
```

Benefit: The bne instruction uses PC-relative addressing mode to loop back efficiently

Drawback: the range is limited, can only branch within $\pm 2^{15}-1$ instruction words from the current PC

2.27.4.

For a. ① 0x3C100064

② 0x3610002B

For b: ① 0x20080000

② 0x8D090400

2.27.5.

```
addi $s0, $zero, 0
 ori $s0, $s0, part_of_data
```

2.27.6.

Typically require 2-3 extra instructions