JOINT INSTITUTE
交大密西根学院

Ve370 Introduction to Computer Organization

Homework 1

1. Exercise 1.3.1 ~ 1.3.3

1.3.1 P2 has the highest performance.

Instr/sec = f/CPI

a.	performance of P1 (instructions/sec) = $3 \times 10^9/1.5 = 2 \times 10^9$ performance of P2 (instructions/sec) = $2.5 \times 10^9/1.0 = 2.5 \times 10^9$ performance of P3 (instructions/sec) = $4 \times 10^9/2.2 = 1.8 \times 10^9$
b.	performance of P1 (instructions/sec) = $2 \times 10^9/1.2 = 1.66 \times 10^9$ performance of P2 (instructions/sec) = $3 \times 10^9/0.8 = 3.75 \times 10^9$ performance of P3 (instructions/sec) = $4 \times 10^9/2 = 2 \times 10^9$

1.3.2 No. cycles = time × clock rate

time = (No. Instr × CPI)/clock rate, then No. instructions = No. cycles/CPI

a.	cycles(P1) = $10 \times 3 \times 10^9 = 30 \times 10^9$ s cycles(P2) = $10 \times 2.5 \times 10^9 = 25 \times 10^9$ s cycles(P3) = $10 \times 4 \times 10^9 = 40 \times 10^9$ s No. instructions(P1) = $30 \times 10^9/1.5 = 20 \times 10^9$ No. instructions(P2) = $25 \times 10^9/1 = 25 \times 10^9$ No. instructions(P3) = $40 \times 10^9/2.2 = 18.18 \times 10^9$
b.	cycles(P1) = $10 \times 2 \times 10^9 = 20 \times 10^9$ s cycles(P2) = $10 \times 3 \times 10^9 = 30 \times 10^9$ s cycles(P3) = $10 \times 4 \times 10^9 = 40 \times 10^9$ s No. instructions(P1) = $20 \times 10^9/1.2 = 16.66 \times 10^9$ No. instructions(P2) = $30 \times 10^9/0.8 = 37.5 \times 10^9$ No. instructions(P3) = $40 \times 10^9/2 = 20 \times 10^9$

1.3.3 $time_{new} = time_{old} \times 0.7 = 7$ s

a.	$CPI_{new} = CPI_{old} \times 1.2$, then CPI(P1) = 1.8, CPI(P2) = 1.2, CPI(P3) = 2.6 *f* = No. Instr × CPI/time, then *f*(P1) = $20 \times 10^9 \times 1.8 / 7 = 5.14$ GHz *f*(P2) = $25 \times 10^9 \times 1.2 / 7 = 4.28$ GHz *f*(P1) = $18.18 \times 10^9 \times 2.6 / 7 = 6.75$ GHz
b.	$CPI_{new} = CPI_{old} \times 1.2$, then CPI(P1) = 1.44, CPI(P2) = 0.96, CPI(P3) = 2.4 *f* = No. Instr × CPI/time, then *f*(P1) = $16.66 \times 10^9 \times 1.44/7 = 3.42$ GHz *f*(P2) = $37.5 \times 10^9 \times 0.96/7 = 5.14$ GHz *f*(P1) = $20 \times 10^9 \times 2.4/7 = 6.85$ GHz

2. Exercise 1.4.4 ~ 1.4.6

1.4.4

a.	$(650 \times 1 + 100 \times 5 + 600 \times 5 + 50 \times 2) \times 0.5 \times 10^{-9} = 2{,}125$ ns
b.	$(750 \times 1 + 250 \times 5 + 500 \times 5 + 500 \times 2) \times 0.5 \times 10^{-9} = 2{,}750$ ns

1.4.5 CPI = time × clock rate/No. instr

a.	CPI $= 2{,}125 \times 10^{-9} \times 2 \times 10^{9}/1{,}400 = 3.03$
b.	CPI $= 2{,}750 \times 10^{-9} \times 2 \times 10^{9}/2{,}000 = 2.75$

1.4.6

a.	Time $= (650 \times 1 + 100 \times 5 + 300 \times 5 + 50 \times 2) \times 0.5 \times 10^{-9} = 1{,}375$ ns Speedup = 2,125 ns/1,375 ns = 1.54 CPI $= 1{,}375 \times 10^{-9} \times 2 \times 10^{9}/1{,}100 = 2.5$
b.	Time $= (750 \times 1 + 250 \times 5 + 250 \times 5 + 500 \times 2) \times 0.5 \times 10^{-9} = 2{,}125$ ns Speedup = 2,750 ns/2,125 ns = 1.29 CPI $= 2{,}125 \times 10^{-9} \times 2 \times 10^{9}/1{,}750 = 2.43$

3. Exercise 1.10.1 ~ 1.10.5

1.10.1

a.	Processors	Instructions per Processor	Total Instructions
	1	4096	4096
	2	2048	4096
	4	1024	4096
	8	512	4096

b.	Processors	Instructions per Processor	Total Instructions
	1	4096	4096
	2	2048	4096
	4	1024	4096
	8	512	4096

1.10.2

a.	Processors	Execution Time (μs)
	1	4.096
	2	2.368
	4	1.504
	8	1.152

b.	Processors	Execution Time (μs)
	1	4.096
	2	2.688
	4	1.664
	8	0.992

1.10.3

a.	Processors	Execution Time (μs)
	1	5.376
	2	3.008
	4	1.824
	8	1.312

b.	Processors	Execution Time (μs)
	1	5.376
	2	3.328
	4	1.984
	8	1.152

1.10.4

a.	Cores	Execution Time (s) @ 3 GHz
	1	4.00
	2	2.33
	4	1.50
	8	1.08

b.	Cores	Execution Time (s) @ 3 GHz
	1	3.33
	2	2.00
	4	1.16
	8	0.71

1.10.5

a.	Cores	Power (W) per Core @ 3 GHz	Power (W) per Core @ 500 MHz	Power (W) @ 3 GHz	Power (W) @ 500 MHz
	1	15	0.625	15	0.625
	2	15	0.625	30	1.25
	4	15	0.625	60	2.5
	8	15	0.625	120	5
b.	**Cores**	**Power (W) per Core @ 3 GHz**	**Power (W) per Core @ 500 MHz**	**Power (W) @ 3 GHz**	**Power (W) @ 500 MHz**
	1	15	0.625	15	0.625
	2	15	0.625	30	1.25
	4	15	0.625	60	2.5
	8	15	0.625	120	5

4. Exercise 2.3.1

2.3.1

a.	`sub f, $0, f` `sub f, f, g`

b.	`sub f, $0, f`
	`addi f, f, -5 (note, no subi)`
	`add f, f, g`

5. Exercise 2.6.1

2.6.1

a.	`lw $t0, 4($s7) # $t0 <-- B[1]`
	`sub $t0, $t0, $s1 # $t0 <-- B[1] - g`
	`add $s0, $t0, $s2 # f <-- B[1] -g + h`

b.	`sll $t0, $s1, 2 # $t0 <-- 4*g`
	`add $t0, $t0, $s7 # $t0 <-- Addr(B[g])`
	`lw $t0, 0($t0) # $t0 <-- B[g]`
	`addi $t0, $t0, 1 # $t0 <-- B[g]+1`
	`sll $t0, $t0, 2 # $t0 <-- 4*(B[g]+1) = Addr(A[B[g]+1])`
	`lw $s0, 0($t0) # f <-- A[B[g]+1]`

6. Exercise 2.6.4

2.6.4

a.	`f = f - i;`
b.	`f = 2 * (&A);`

7. Exercise

Solution:

000000 00000 01000 01010 00000 101010
000101 01010 00000 00000 00000 000001
000010 00000 00000 00000 00100 000110
001000 10010 10010 00000 00000 000010
001001 01000 01000 00000 00000 000001
000010 00000 00000 00000 00100 000000

Note that the sequence of \$rt, \$rs and \$rd is not the same in between machine code and MIPS.

8. Exercise

   lui \$t0, 4096
   lb \$s1, 2(\$t0)

   \$s1: 0x00000089

9. Exercise 2.19.1 (15 points)

a.

```
fib:     addi $sp, $sp, -12        # make room on stack
         sw   $ra, 8($sp)          # push $ra
         sw   $s0, 4($sp)          # push $s0
         sw   $a0, 0($sp)          # push $a0 (N)
         bgt  $a0, $0, test2       # if n>0, test if n=1
         add  $v0, $0, $0          # else fib(0) = 0
         j rtn                     #
test2:   addi $t0, $0, 1           #
         bne  $t0, $a0, gen        # if n>1, gen
         add  $v0, $0, $t0         # else fib(1) = 1
         j rtn
gen:     subi $a0, $a0,1           # n-1
         jal  fib                  # call fib(n-1)
         add  $s0, $v0, $0         # copy fib(n-1)
         sub  $a0, $a0,1           # n-2
         jal  fib                  # call fib(n-2)
         add  $v0, $v0, $s0        # fib(n-1)+fib(n-2)
rtn:     lw   $a0, 0($sp)          # pop $a0
         lw   $s0, 4($sp)          # pop $s0
         lw   $ra, 8($sp)          # pop $ra
         addi $sp, $sp, 12         # restore sp
         jr   $ra

# fib(0) = 12 instructions, fib(1) = 14 instructions,
# fib(N) = 26 + 18N instructions for N >=2
```

b.

```
positive:
       addi $sp, $sp, -4
       sw   $ra, 0($sp)
       jal  addit
       addi $t1, $0, 1
       slt  $t2, $0, $v0
       bne  $t2, $0, exit
       addi $t1, $0, $0
exit:
       add  $v0, $t1, $0
       lw   $ra, 0($sp)
       addi $sp, $sp, 4
       jr   $ra
addit:
       add  $v0, $a0, $a1
       jr   $ra
# 13  instructions worst-case
```

10. Exercise 2.19.2 (10 points)

11. Exercise 2.19.3 (10 points)

2.19.2

a.	Due to the recursive nature of the code, not possible for the compiler to in-line the function call.

b.

```
positive:
        add  $t0, $a0, $a1
        addi $v0, $0, 1
        slt  $t2, $0, $t0
        bne  $t2, $0, exit
        addi $v0, $0, $0
exit:
        jr   $ra

# 6  instructions worst-case
```

2.19.3

a.

```
after calling function fib:
old $sp ->      0x7ffffffc      ???
                 -4             contents of register $ra for fib(N)
                 -8             contents of register $s0 for fib(N)
$sp->           -12             contents of register $a0 for fib(N)

there will be N-1 copies of $ra, $s0, and $a0
```

b.

```
after calling function positive:
old $sp ->      0x7ffffffc      ???
$sp->           -4              contents of register $ra

after calling function addit:
old $sp ->      0x7ffffffc      ???
                -4              contents of register $ra
$sp->           -8              contents of register $ra    #return to
positive
```

12. Exercise 2.23.1

2.23.1

a.
```
MAIN:   addi $sp, $sp, -4
        sw   $ra, ($sp)
        add  $t6, $0, 0x30   # '0'
        add  $t7, $0, 0x39   # '9'
        add  $s0, $0, $0
        add  $t0, $a0, $0

LOOP:   lb   $t1, ($t0)
        slt  $t2, $t1, $t6
        bne  $t2, $0, DONE
        slt  $t2, $t7, $t1
        bne  $t2, $0, DONE
        sub  $t1, $t1, $t6
        beq  $s0, $0, FIRST
        mul  $s0, $s0, 10
FIRST:  add  $s0, $s0, $t1
        addi $t0, $t0, 1
        j LOOP

DONE:   add  $v0, $s0, $0
        lw   $ra, ($sp)
        addi $sp, $sp, 4
        jr   $ra
```

b.
```
MAIN:   addi $sp, $sp, -4
        sw   $ra, ($sp)
        add  $t4, $0, 0x41   # 'A'
        add  $t5, $0, 0x46   # 'F'
        add  $t6, $0, 0x30   # '0'
        add  $t7, $0, 0x39   # '9'
        add  $s0, $0, $0
        add  $t0, $a0, $0

LOOP:   lb   $t1, ($t0)
        slt  $t2, $t1, $t6
        bne  $t2, $0, DONE
        slt  $t2, $t7, $t1
        bne  $t2, $0, HEX
        sub  $t1, $t1, $t6
        j DEC
HEX:    slt  $t2, $t1, $t4
        bne  $t2, $0, DONE
        slt  $t2, $t5, $t1
        bne  $t2, $0, DONE
        sub  $t1, $t1, $t4
        addi $t1, $t1, 10
DEC:    beq  $s0, $0, FIRST
        mul  $s0, $s0, 10
FIRST:  add  $s0, $s0, $t1
        addi $t0, $t0, 1
        j LOOP

DONE:   add  $v0, $s0, $0
        lw   $ra, ($sp)
        addi $sp, $sp, 4
        jr   $ra
```

13. Exercise 2.24.3

2.24.3

a.	0x00000011
b.	0x00115555

14. Exercise 2.26.1

15. Exercise 2.26.2

2.26.1 Branch range is 0x00020000 to 0xFFFE0004.

a.	one branch
b.	one branch

2.26.2

a.	one
b.	can't be done