HW2 Solutions (CS552 Spring 2013)

Grade distribution: (Total: 100)
- Verilog submission of problems 1) and 2) carry 25 points each(Total: 50 points)
- Written part of problem 1) and 2) carry 5 points (Total: 10 points)
- Written part of problems 3) through 14) (only four are graded) carry 10 points each (Total: 40 points)

Grading of verilog submission:
Total points for each problem :25
- Points for compiling design : 5
- Points for functional tests: 20 (1 point deducted for each failure out of 50 random tests)
- Penalty for incorrect directory structure: -5
- Penalty for missing files: (50% of points scored for the problem)
- Penalty for vcheck failures: (50% of points scored for the problem)

Grading of written part:
-written part of problems 1) and 2) – 5 points each
-Problems 3), 5), 8) and 9) – 10 points each

Problem 1 and 2

Solutions not to be provided

Problem 3 (1.3.1 – 1.3.3)

1.3.1 a) perf of P1 = 3G/1.5 =2 X 10^9 instructions per cycle
perf of P2 = 2.5G/1 = 2.5 X 10^9 instructions per cycle
perf of P3 = 4G/2.2 = 1.82 X 10^9 instructions per cycle
P2 is best.

b) perf of P1 = 2G/1.2 =1.67 X 10^9 instructions per cycle
perf of P2 = 3G/0.8 = 3.75 X 10^9 instructions per cycle
perf of P3 = 4G/2 = 2 X 10^9 instructions per cycle
P2 is best.

1.3.2 a) cycles for P1 = 3G X 10 = 3 X 10^{10} cycles
instructions for P1 = 3G X 10 /1.5 = 2 X 10^{10} instructions
cycles for P2 = 2.5G X 10 = 2.5 X 10^{10} cycles
instructions for P2 = 2.5G X 10 /1 = 2.5 X 10^{10} instructions
cycles for P3 = 4G X 10 = 4 X 10^{10} cycles
instructions for P3 = 4G X 10 /2.2 = 1.82 X 10^{10} instructions

b) cycles for P1 = 2G X 10 = 2 X 10^{10} cycles
instructions for P1 = 2G X 10 /1.2 = 1.67 X 10^{10} instructions
cycles for P2 = 3G X 10 = 3 X 10^{10} cycles
instructions for P2 = 3G X 10 /0.8 = 3.75 X 10^{10} instructions
cycles for P3 = 4G X 10 = 4 X 10^{10} cycles
instructions for P3 = 4G X 10 /2 = 2 X 10^{10} instructions

1.3.3 new time = 10 X 0.7 = 7s

a) CPInew = CPIold × 1.2, then CPI(P1) = 1.8, CPI(P2) = 1.2, CPI(P3) = 2.6
f = No. Instr × CPI/time, then
f(P1) = 20 × 10^9 × 1.8 / 7 = 5.14 GHz
f(P2) = 25 × 10^9 × 1.2 / 7 = 4.28 GHz
f(P1) = 18.18 × 10^9 × 2.6 / 7 = 6.75 GHz

b) CPInew = CPIold × 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

Problem 4 (1.4.1 – 1.4.3)

1.4.1

Class A: 105 instr.
Class B: 2 × 105 instr.
Class C: 5 × 105 instr.
Class D: 2 × 105 instr.
Time = No. instr × CPI/clock rate
a)Total time P1 = $(10^5 + 2 \times 10^5 \times 2 + 5 \times 10^5 \times 3 + 2 \times 10^5 \times 3)/(2.5 \times 10^9) = 10.4 \times 10^{-4}$ s
b)Total time P2 = $(10^5 \times 2 + 2 \times 10^5 \times 2 + 5 \times 10^5 \times 2 + 2 \times 10^5 \times 2)/(3 \times 10^9) = 6.66 \times 10^{-4}$ s

1.4.2 CPI = time × clock rate/No. instr

a) CPI (P1) = $10.4 \times 10^{-4} \times 2.5 \times 10^9/10^6 = 2.6$
CPI (P2) = $6.66 \times 10^{-4} \times 3 \times 10^9/10^6 = 2.0$

b) CPI (P1) = $6.8 \times 10^{-4} \times 2.5 \times 10^9/10^6 = 1.7$
CPI (P2) = $4 \times 10^{-4} \times 3 \times 10^9/10^6 = 1.2$

1.4.3

a) clock cycles (P1) = $10^5 \times 1 + 2 \times 10^5 \times 2 + 5 \times 10^5 \times 3 + 2 \times 10^5 \times 3 = 26 \times 10^5$
clock cycles (P2) = $10^5 \times 2 + 2 \times 10^5 \times 2 + 5 \times 10^5 \times 2 + 2 \times 10^5 \times 2 = 20 \times 10^5$

b) clock cycles (P1) = 17×10^5
clock cycles (P2) = 12×10^5

Problem 5 (1.4.4 – 1.4.6)

1.4.4 a) $(650 \times 1 + 100 \times 5 + 600 \times 5 + 50 \times 2) / (2 \text{ X } 10^9) = 2{,}125$ ns
b) $(750 \times 1 + 250 \times 5 + 500 \times 5 + 500 \times 2) / (2 \times 10^9) = 2{,}750$ 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 = $(650 \times 1 + 100 \times 5 + 300 \times 5 + 50 \times 2) / (2 \text{ X } 10^9) = 1{,}375$ 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 = $(750 \times 1 + 250 \times 5 + 250 \times 5 + 500 \times 2) / (2 \text{ X } 10^9) = 2{,}125$ ns
Speedup = 2,750 ns/2,125 ns = 1.29
CPI = $2{,}125 \times 10^{-9} \times 2 \times 10^9/1{,}750 = 2.43$

Problem 6 (1.6.1 – 1.6.3)

1.6.1 CPI = $T_{exec} \times f$/No. Instr

a) CPI for compiler A = $1.8 \text{ X } 10^9 / 10^9 = 1.8$
CP for compiler B = $1.8 \text{ X } 10^9 / (1.2 \text{ X } 10^9) = 1.5$

b) CPI for compiler A = $1.1 \text{ X } 10^9 / 10^9 = 1.1$
CP for compiler B = $1.5 \text{ X } 10^9 / (1.2 \text{ X } 10^9) = 1.25$

1.6.2 f_A/f_B = (No. Instr(A) X CPI(A))/(No. Instr(B) X CPI(B))

a) $f_A/f_B = (1 \text{ X } 10^9 \text{ X } 1.8)/(1.2 \text{ X } 10^9 \text{ X } 1.5) = 1.00$
b) $f_A/f_B = (1 \text{ X } 10^9 \text{ X } 1.1)/(1.2 \text{ X } 10^9 \text{ X } 1.25) = 0.73$

1.6.3 T_{new}/T_{old} = (I_{count}(new) X CPI(new)) / (I_{count}(old) X CPI(old))

a) $T_{new}/T_A = (0.6 \text{ X } 10^9 \text{ X } 1.1) / (1 \text{ X } 10^9 \text{ X } 1.8) = 0.37$
$T_{new}/T_B = (0.6 \text{ X } 10^9 \text{ X } 1.1) / (1.2 \text{ X } 10^9 \text{ X } 1.5) = 0.37$

a) $T_{new} / T_A = (0.6 \times 10^9 \times 1.1) / (1 \times 10^9 \times 1.1) = 0.6$

$T_{new} / T_B = (0.6 \times 10^9 \times 1.1) / (1.2 \times 10^9 \times 1.25) = 0.44$

Problem 7 (Only part A of 1.10.1 – 1.10.3)

1.10.1 For 1 processor system, instruction per processor = 4096, Total instructions = 4096
For 1 processor system, instruction per processor = 2046 Total instructions = 4096
For 1 processor system, instruction per processor = 1028, Total instructions = 4096
For 1 processor system, instruction per processor = 512, Total instructions = 4096

1.10.2 $T = [(I_{count} \text{ (Arith) X CPI (Arith)}) + (I_{count} \text{ (ldst) X CPI (ldst)}) + (I_{count} \text{ (branch) X CPI (branch)})]/f$
For 1 processor, $T = (2560 \times 1 + 1280 \times 4 + 256 \times 2) / 2 \times 10^9 =$ 4.096 microseconds
For 2 processor, $T = (1280 \times 1 + 640 \times 5 + 128 \times 2) / 2 \times 10^9 =$ 2.368 microseconds
For 4 processor, $T = (640 \times 1 + 320 \times 7 + 64 \times 2) / 2 \times 10^9 =$ 1.504 microseconds
For 8 processor, $T = (320 \times 1 + 160 \times 12 + 32 \times 2) / 2 \times 10^9 =$ 1.152 microseconds

1.10.3 For 1 processor, $T = (2560 \times 2 + 1280 \times 4 + 256 \times 2) / 2 \times 10^9 =$ 5.376 microseconds
For 2 processor, $T = (1280 \times 2 + 640 \times 5 + 128 \times 2) / 2 \times 10^9 =$ 3.008 microseconds
For 4 processor, $T = (640 \times 2 + 320 \times 7 + 64 \times 2) / 2 \times 10^9 =$ 1.824 microseconds
For 8 processor, $T = (320 \times 2 + 160 \times 12 + 32 \times 2) / 2 \times 10^9 =$ 1.312 microseconds

Problem 8 (2.10.1 – 2.10.3)

2.10.1 a) add $s0, $s0, $s0
b) sub $t1, $t2, $t3

2.10.2 a) R type
b) R type

2.10.3 a) 0x02108020
b) 0x014B4822

Problem 9

sub $t1, $t2, $t3
add $t4, $t1, $t3
sub $t2, $t1, $4

Problem 10 (2.13.1 – 2.13.3)(Use modified instructions for 2.13.1)

2.13.1 a) 0xBABEFEF8
b) 0x11DD11D1

2.13.2 a) 0xAAAAAAA0
b) 0x00DD00D0

2.13.3 a) 0x00005545
b) 0x0000BA01

Problem 11(Only part B of 2.14.1 – 2.14.3)

2.14.1
```
lui $t1, 0x003f
ori $t1, $t1, 0xffe0
and $t1, $t0, $t1
sll $t1, $t1, 9
```

2.14.2
```
andi $t0, $t0, 0x000f
sll $t0, $t0, 14
ori $t1, $t1, 0x3fff
sll $t1, $t1, 18
ori $t1, $t1, 0x3fff
or $t1, $t1, $t0
```

2.14.3
```
srl $t0, $t0, 28
andi $t0, $t0, 0x0007
sll $t0, $t0, 14
ori $t1, $t1, 0x7fff
```

```
sll $t1, $t1, 17
ori $t1, $t1, 0x3fff
or $t1, $t1, $t0
```

Problem 12

```
add $t2, $zero, 10                  # i=10
loopstart: beq $t2, $zero, loopdone # Jump to end of loop if i == 0
sll $t3, $t2, 4                     # $t3 = 16 * i
add $t3, $a0, $t3                   # $t3 = address of a[4 * i]
sll $t4, $t2, 3                     # $t4 =  8 * i
add $t4, $a1, $t4                   # $t4 = address of b[2 * i]
lw $t4, 0($t4)                      # $t4 = b[2 * i]
add $t4, $t4, $t2                   # $t4 = b[2 * i]  + i
sw $t4, 0($t3)                      # a[4 * i] = $t4
sub $t2, $t2, 1                     # i--
j loopstart                         # Jump to beginning of loop
loopdone: ...
```

Problem 13 (2.39.1 – 2.39.3)

2.39.1 T = [(CPI(arith) X I_{count}(arith)) + (CPI(ldst) X I_{count}(ldst)) + (CPI(branch) X I_{count}(branch))] / f

a) T = (1 X 500M + 10 X 300M + 3 X 100M) / 5G = 0.76s

b) T = (4 X 500M + 40 X 300M + 3 X 100M) / 5G = 2.86s

2.39.2 a) $T_{\textbf{new}}$ = (1 X **375**M + 10 X 300M + 3 X 100M) / (5G/**1.1**) = 0.81s

ie., the extra clock cycle time adds sufficiently to the new CPU time such that it is not quicker than the old execution time.

b) $T_{\textbf{new}}$ = (4 X **375**M + 40 X 300M + 3 X 100M) / (5G/**1.1**) = 3.04s

ie., the extra clock cycle time adds sufficiently to the new CPU time such that it is not quicker than the old execution time.

2.39.3 a) CPI_{old} = (1 X 500M +10 X 300M + 3 X 100M) / 900M = 4.22

CPI_{new} (for doubling arith perf) = (0.5 X 500M +10 X 300M + 3 X 100M) / 900M = 3.94

Speedup(for doubling arith perf) = 4.22/3.94 = 107%

CPI_{new} (for 10x arith perf) = (0.1 X 500M +10 X 300M + 3 X 100M) / 900M = 3.72

Speedup(for 10x arith perf) = 4.22/3.72 = 113%

b) CPI_{old} = (4 X 500M +40 X 300M + 3 X 100M) / 900M = 15.89

CPI_{new} (for doubling arith perf) = (2 X 500M +40 X 300M + 3 X 100M) / 900M = 14.78

Speedup(for doubling arith perf) = 15.89/14.78 = 107%

CPI_{new} (for 10x arith perf) = (0.4 X 500M +40 X 300M + 3 X 100M) / 900M = 13.89

Speedup(for 10x arith perf) = 15.89/13.89 = 114%

Problem 14

objdump is a program for displaying various information about object files. For instance, it can be used as a disassembler to view executable in assembly form. It is part of the GNU binutils for fine-grained control over executable and other binary data.

```
400474    push %rbp
400475    mov %rsp,%rbp             //Setting up stack registers
400478    movl $0x0,-0x4(%rbp)      //Initialise sum, -0x4(%rbp)
```

```
40047f    movl $0x0,-0x8(%rbp)        //Initialize i
400486    jmp 400492 <main+0x1e>      //check the loop condition before first iteration
400488    mov -0x8(%rbp),%eax         //load value of i into register %eax
40048b    add %eax, -0x4(%rbp)        //add i to sum
40048e    addl $0x1,-0x8(%rbp)        //increment i
400492    cmpl $0x17, -0x8(%rbp)      //compare i to 23 (ie, 0x17)
400496    jle 400488 <main+0x14>      //if i<=23, go to beginning of loop
400498    mov -0x4(%rbp), %eax        //Keep value of sum in %eax before returning
40049b    leaveq
40049c    retq
40049d    nop
40049e    nop
40049f    nop
```

With -O3:

```
mov    $0x114, %eax
retq
```

The compiler optimizes the loop and determines the sum without executing the loop. Note that 0x114 or 276 in decimal is the sum of integers from 0 to 23!

The other instructions set up context to start the program and are part of standard libraries to load a program.