Siddhesh Kanawade
20110199

Indian Institute of Technology Gandhinagar

ES215: COA

ASSIGNMENT 2

[Link to the github repo]

Assignment 2

1) Here I have assumed that the array consists of **integer values**
   MIPS Assembly code:

```
            addi $gp, $0, 200
            lw $s0, 0($gp) #max=A[0]
            add $s1, $0, $0 #max-index=0
            lw $s2, 0($gp)          #min=A[0]
            add $s3, $0, $0 #min-index=0
            add $s4, $0, $0 #average=0
            add $t0, $0, $0 #i, for loop
            addi $t1, $0, 101 #n, size
for:    beq $t0, $t1, e
            sll $t4, $t0, 2
            add $t5, $t4, $gp
            slt $t2, $s0, $t5
            bne $t2, $0, max
            j a
max:    lw $s0, $t5
            lw $s1, $t0
a:          sgt $t3, $s2, $t5
            bne $t3, $0, min
            j plus
min:    lw $s2, $t5
            lw $s3, $t0
plus:   add $s4, $s4, $t5
            addi $t0, $t0, 1
            j for
e:          div $s4, $t1
            mflo $s4
            jr $ra
```

---

2) Dual core processor
   Program A completes in 6 seconds on core 1 and has a CPI of 6.
   Program B completes in 5 seconds on core 2 and has a CPI of 5.
   Cores run at 1 GHz.

   Execution time = $(CPI\ *\ IC)/\ CLOCK\ RATE$
   Instruction count = $Execution\ time\ \ *\ \ Clock\ Rate\ /\ CPI$

Assume the total combined execution time = x sec.

Instruction count in Program 1 = $10^9 * x/6$
Instruction count in Program 2 = $10^9 * x/5$

Total IC = $Instruction\ count\ in\ program\ 1 + Instruction\ count\ in\ program\ 2$
= $10^9 * x * ((1/5) + (1/6))$

Now,
Combined Throughput = $Total\ IC / Execution\ time$
= $10^9 * x * ((1/5) + (1/6))/x$
= $10^9 * 11/30$
= $0.366667 * 10^9$ instructions per seconds

---

3) X => 2 GHz
Y => 4 GHz
Program A, X => 10 billion instructions
=> avg. CPI of 3.
Program A, Y => 7 billion instructions
=> avg. CPI of 5.

Now, calculating:
Running time =$Instruction\ Count * avg.\ CPI / Clock\ Rate$
Program running time on processor X=$10 * 10^9 * 3/(2 * 10^9)$ = $15\ sec$
Program running time on processor Y= $7 * 10^9 * 5/(4 * 10^9)$ = $8.75\ sec$

Speedup of program A on processor Y over X = $Running\ time\ on\ X/Running\ time\ on\ Y$
= $15/8.75$
= 1.7142857

---

4) Now,

Execution Time = $Instruction\ Count * CPI / Clock\ Rate$
IC = 9 billion
Avg CPI = 1.5
Execution time on old processor = 9 *10^9 * 1.5 / 10^9
= $13.5\ seconds$
Here,
Execution time on new processor is quarter times the old
Clock Rate = 2 GHz

Assumption: The Instruction count remains the same.

$13.5/4 = 9 * 10^8 * avg.\ CPI/(2 * 10^9)$

Therefore,

$Avg.\ CPI = 0.75$

---

5)

Total Power = 80 watts
Clock Rate = 2*10^9 Hz
Operating Voltage (V) = 5 V
Here,

Total Power = $Static\ Power + Dynamic\ Power$
Dynamic Power = $(½) * C * V^2 * f$
Static Power = $I * V$

a) Static power is 40% of the total power.
Thus, the standard condition

$P_{static} = 40\% * Total\ Power$
$= 32\ watts$

$P_{dynamic} = 80 - 32$
$= 48\ watts$

b) Static Power is proportional to voltage and Dynamic Power is directly proportional to the square of the voltage.
Now,

At 2 volts,
$Pstatic = 32 * 2 / 5 = 12.8\ watts$

$Dynamic\ Power = 48 * 4 / 25 = 7.68\ watts$

$Total\ Power\ Consumption = 12.8 + 7.68 = 20.48\ watts$

$Fraction\ of\ Static\ Power = Pstatic/Total\ Power = 12.8 / 20.48 = 0.625$

Grace Question

a) Language used: **C**
   Code used: **fib_loop.c**

   Here I have used a cross compiler toolchain to compile **mips** code.
   There are two versions of mips code: **MIPS32 and MIPS64**

   Commands used:
   x_86: gcc fib_loop.c -o fib_loop_x86.out
   MIPS32: mips-openwrt-linux-gcc fib_loop.c -o fib_loop_mips.out
   MIPS64: mips-openwrt-linux-gcc fib_loop.c -o fib_loop_mips64.out -march=mips64r2

```
siddhesh@siddhesh-G3-3500:~/Desktop/Siddhesh/Academics/COA Assignment/ES215-COA_Assignments/Assignment 2/Grace Question$ file fib_loop_mips64.out
fib_loop_mips64.out: ELF 32-bit MSB executable, MIPS, MIPS64 rel2 version 1 (SYSV), dynamically linked, interpreter /lib/ld-musl-mips-sf.so.1, with debug_info, not stripped
```

```
siddhesh@siddhesh-G3-3500:~/Desktop/Siddhesh/Academics/COA Assignment/ES215-COA_Assignments/Assignment 2/Grace Question/mips$ file fib_loop_mips.out
fib_loop_mips.out: ELF 32-bit MSB executable, MIPS, MIPS32 rel2 version 1 (SYSV), dynamically linked, interpreter /lib/ld-musl-mips-sf.so.1, with debug_info, not stripped
```

b) Individual file sizes

File type	x_86	MIPS32	MIPS64
.i	23.5 kb	10.4 kb	10.4 kb
.s	2.5 kb	2.9 kb	2.8 kb
.obj	2.7 kb	2.2 kb	2.2 kb
.out	16.9 kb	8.4 kb	8.3 kb

Observations:

1. The size of the preprocessed file is more than double in case of x_86 architecture.
2. The size of compiled code, and assembled code is comparable and somewhat similar. To be precise, the size of compiled code is larger in MIPS and size of assembled code is larger in x_86 architecture.
3. The size of binary code/executable file is more in case of x_86 as compared to MIPS.

Other observations:

All the above files are dynamically linked, in case of statically linked files the size of files increases drastically. To make static files, the following command has to be used.

mips-openwrt-linux-gcc fib_loop.c -o fib_loop_mips64.out -static -march=mips64r2