**COMPUTER ARCHITECTURE & ASSEMBLY LANGUAGE**
**14:332:331**
**Rutgers University**
**Fall 2016**

Solution of Homework 1
Due: Sep.30, 2016

1. You are on the design team for a new processor. The clock of the processor runs at 500 MHz. The following table gives instruction frequencies for Benchmark B, as well as how many cycles the instructions take, for the different classes of instructions. For this problem, we assume that (unlike many of today's computers) the processor only executes one instruction at a time.

| Instruction Type | Frequency | Cycles |
|---|---|---|
| **Loads & Stores** | 30% | 5 cycles |
| **Arithmetic Instructions** | 40% | 6 cycles |
| **All Others** | 30% | 3 cycles |

   a. Calculate the CPI for Benchmark B.

   CPI= (0.30 * 5) + (0.40 * 6) + (0.30 * 3) = 4.8

   b. The CPU execution time on the benchmark is exactly 11 seconds. What is the native "MIPS" processor speed for the benchmark in millions of instructions per second?

   The formula for calculating MIPS is:

   CPI=Clock per instruction =4.8

   ➔ One instruction needs 4.8 clock cycle=$4.8*(1/(500*10^6))$ second=$0.0096*10^{-6}$ s

   ➔ The number of instructions that run in 1 second=$1/(0.0096*10^{-6})=104.16*10^{+6}$

   ➔ MIPS (Mega instructions per second)=104.16

   c. The hardware expert says that if you double the number of registers, the cycle time must be increased by 20%. What would the new clock speed be (in MHz)?

   Clock frequency = 1/Clock Cycle Time
   Clock Cycle Time = 1/Clock frequency
   Cycle Time = $1/(500 * 10^6) = 2 * 10^{-9}$
   The cycle time is then increased by 20%:
   $(2 * 10^{-9}) * 1.2 = 2.4 * 10^{-9}$
   The new clock rate is thus:
   $1/(2.4 * 10^{-9})$ = 416.667 * 106 or 416.667 MHz

d. The compiler expert says that if you double the number of registers, then the compiler will generate code that requires only half the number of Loads & Stores. What would the new CPI be on the benchmark?

Assume that we had 100 instructions in part b, so we will reduce the number of loads and stores by half, and this will reduce the total number of instructions. So the new instruction mix will be:
15 Loads and Stores
40 Arithmetic Instructions
30 All Others

The total number of instructions is now 85, so the answer is:
(CPI= (15 * 5) + (40 * 6) + (30 * 3)) / 85 = 405 cycles/ 85 instructions = 4.76

e. How many CPU seconds will the benchmark take if we double the number of registers (taking into account both changes described above)?

CPU seconds = (Number of instructions * Number of Clocks per instructions)/Clock Rate

First thing we need to do, is to calculate the number of instructions which execute in 11 seconds on the new benchmark - the one with half the number of loads and stores. To do this, we will need to figure out how many instructions execute on the original benchmark in 11 seconds.

CPU Time=IC*CPI*CCT → $11=IC*4.8*1/(500 * 10^6)$ → $IC\ (original)=1146*10^6$

Now we need to figure out how many of those are Loads and Stores in the original benchmark so:

$(1146 * 10^6) * 0.3 = 343.8 * 10^6$ are Load and Store instructions because the chart says that 30% of all instructions are Loads and Stores. Now we need to cut this number in half, because the new benchmark says that we have half the number of loads and stores. So in the new benchmark we have $1146*10^6 - 343.8 * 10^6 = 802.2*10^6$ instructions
Also the cycle time increases by 20%. Therefore the final solution is:
CPU Time (new)= $802.2*10^6 *4.76*2.4 * 10^{-9} = 9.164$ seconds

2. Assume a program includes 3000 integer instructions (INT), 30,000 floating point (FP) instructions, 5000 Load/Store (L/S) instructions and 10000 branch instructions. The CPI for each type of instruction is 1, 1, 3 and 2, respectively. Assume that the processor has a 2.6 GHz clock rate.

   a. By how much must we improve the CPI of FP instructions if we want the program to run 50% faster?
   b. By how much must we improve the CPI of L/S instruction if we want the program to run 30% faster?

c. By how much is the execution time of the program improved if the CPI of INT and FP instructions are reduced by 25% and the CPI of L/S and Branch is reduced by 30%?

Sol)

Total instruction count = 3000+30000+5000+10000 = 48000

% of INT instructions = 3000 / 48000 = 1/16
% of FP instructions = 30000/ 48000 = 5/8
% of L/S instructions = 5000/48000 = 5/48
% of Branch instructions =10000/48000 = 5/24

CPI original = $CPI_{FP}$ *%of FP Instr +$CP_{IINT}$ *%of INT Instr +$CPI_{L/S}$ *%of L/S Instr + $CPI_{BR}$ * % of Branch Instr = 17/12

a)

$CPI_{original}/CPI_{new}$=1.5,
$CPI_{new}$ = 17/12 / 1.5 = 17/18
$CPI_{new}$ =17/18 = $CPI_{FP\_new}$ * %of FP Instr +$CPI_{INT}$ *%of INT Instr +$CPI_{L/S}$ *%of L/S Instr + $CPI_{BR}$ * % of Branch Instr
17/18 = 1* 1/16 + $CPI_{FP\_new}$ *5/8 + 3 * 5/48 + 2 * 5/24
$CPI_{FP\_new}$ = 11/45

b) $CPI_{new}$ = 17/12 / 1.3 = 85/78
CPI_new = 85/78 = $CPI_{FP}$ *%of FP Instr +$CP_{IINT}$ *%of INT Instr +$CPI_{L/S\ new}$ *%of L/S Instr + $CPI_{BR}$ * % of Branch Instr
85/78 = 1 * 1/16 + 1 * 5/8 + CPI_L/S new * 5/48 + 2 * 5/24
CPI_L/S new <0
Impossible.

c) CPI_int new = 0.75, CPI_fp new = 0.75
CPI_L/S new = 2.1, CPI_Br new = 1.4
CPI new = $CPI_{FP}$ *%of FP Instr +$CP_{IINT}$ *%of INT Instr +$CPI_{L/S}$ *%of L/S Instr + $CPI_{BR}$ * % of Branch Instr = 0.75*1/16 +0.75*5/8 +2.1*5/48+1.4*5/24 = 197/192
Improvement = CPI original / CPI new = (17/12)/(197/192) = 1.38 times

3. Suppose you have a machine which executes a program consisting of 50% floating point multiply, 20% floating point divide, and the remaining 30% are from other instructions.
   a. Management wants the machine to run 4 times faster. You can make the divide run at most 3 times faster and the multiply run at most 8 times faster. Can you meet management's goal by making only one improvement, and which one?
   b. If you make both the multiply and divide improvements, what is the speed of the improved machine relative to the original machine?

a.

Exe_time (old)= CCT*IC* (0.20*CPI+0.50*CPI+0.30*CPI)=CCT*IC*CPI*1

EXE_time(new when only multiply improves)= CCT*IC*(0.20*CPI+0.50*CPI/8+0.30*CPI)= CCT*IC*CPI*0.5625

➔ Speedup (only multiply improves)=1/0.5625=1.77

EXE_time(new when only divide improves)= CCT*IC*(0.20/3*CPI+0.50*CPI+0.30*CPI)= CCT*IC*CPI*0.8666

➔ Speedup (only divide improves)=1/0.8666=1.15

So by none of these improvements, the goal is achieved.

b.

EXE_time(new)= CCT*IC*(0.20*CPI/3+0.50*CPI/8+0.30*CPI)= CCT*IC*CPI*0.4285

➔ Speedup=1/0.4285=2.33

4. The base address of the array B is stored in $s2. We want to load B[16] into $t1. In the following cases, please fill the blanks with appropriate parameters to do so (Please show the values in **HEX**)

   a. **lw $t1, 64 ($s2) //4*16=64 → In HEX=40**

   b. **addi $t0, $s2, 24**
      **lw $t1, 40($t0) //64-24= 40 → In HEX=28**

   c. **addi $t0, $s2, 96**
      **lw $t1, -32($t0) //64-96=-32 → In HEX =FFFFFFE0**

5. The following instructions are pseudo instructions that are not included in the MIPS ISA. Please write down the shortest sequence of MIPS instructions that perform the same operation.
   a. **subi $t1, $t4, 10** #$t1=$t4-10
   b. **subi $t4, $t2, $2^{22}$** #$t4=$t2-$2^{22}$
   c. **rpt $t1, L1** #if ($t1<0) $t1=$t1+1, go to L1

   **Sol)**
   a) addi $t1, $t4, -10

   b) addi $t1, $0, 2 //$t1=2
      sll $t1,$t1, 21 //$t1=2^22
      sub $t4, $t2, $t1 //$t4=$t2 - $t1=$t2 - 2^22

   c) slt $t0, $t1, $0 //$t0=1, if $t1<0
      beq $t0, $0, OUT //if $t0 = 0, goto OUT
      addi $t1, $t1, 1
      j L1
   OUT:

6. Find the shortest sequence of MIPS instruction that extracts bits 22 down to 4 from register $t0 and compute the half of this value (the value is assumed unsigned integer) and use the new value to replace bits 24 down to 6 in register $t1 (note that other bits of $t1 will remain unchanged).

```
sll $t0,$t0,9 //push out 9 most significant bits
srl $t0,$t0,14 //push out the 4 most significant bits of the initial number and divide by 2
sll $t0,$t0,6 // move the number to position to occupy bits 24 down to 6
lui $t3, 0x FE00
ori $t3,$t3,0x 003F
and $t1,$t1,$t3 //Reset bits 24 to 6 to 0, let all others be what they were before
or $t1,$t1,$t2
```

7. Compile the assembly code for the following C codes. Assume that i, j, and k have been stored in $s0, $s1, and $s2 respectively. The base address of the array B is stored in $s4. Please only use only **TRUE** MIPS instructions.

a. **for (i=k; i>0; i=i-2)**
   **j = i +2k;**

b. **for (i=0; i<k; i++)**
   **for (j=1; j<i; j++)**
   **B[2j]=B[i-k]-4**

**Sol)**

```
a)      add $s0, $0, $2
  LOOP: beq $s0, $0, EXIT
        slt $t1,$s0, $0
        bne $t1, $0, EXIT
        sll $t0, $s2, 1
        add $s1, $s0, $t0
        addi $s0, $s0, -2
        j LOOP
EXIT:   ...
```

```
b)      add $s0, $0, $0         //i=0
        addi $s1, $0, 1         //j=1
LOOPI:  slt $t0,$s0,$s2
        beq $t0, $zero, EXITI
LOOPJ:  slt $t1,$s1,$s0
        beq $t1, $zero, EXITJ
        sub $t2, $s0, $s2       //$t0=i-k
        sll $t2,$t2,2           //4*(i-j)
        add $t2, $t2,$s4
        lw $t3,0($t2)           //$t3=B[i-k]
        addi $t3, $t3, -4       //$t1=B[j-4i]
        sll $t4, $s1, 3         //$t1=4*(2j)
        add $t4,$t4,$s4
        sw $t3, 0($t4)
        addi $s1, $s1, 1
```

```
        J LOOPJ
EXITJ:
        addi $s0, $s0, 1
        J LOOPI
EXITI:
        …
```