# CSE 490/590 Spring 2023 Homework 1 Solution

1) A program's run time is determined by the product of instructions per program, cycles per instruction, and clock frequency. Assume the following instruction mix for a MlPS-like instruction set on the right, Compute the overall CPI

| Instructions | % | cycle |
|---|---|---|
| store and load | 40% | 2 |
| branch | 10% | 4 |
| integer ALU | 30% | 1 |
| shift | 10% | 1 |
| integer multiplies | 10% | 10 |

CPI = (0.40*2)+(0.10*4)+(0.30*1)+(0.10*1)+(0.10*10) = 2.6

The calculation performed is a weighted average of of each of the individual types of the instructions that are present. In this question we assume that the CPU is a single cycle processor, thus each instruction will execute one after the other with no overlap. Each instruction takes a different number of cycles to fully execute.

2) Consider three different processors P1, P2, and P3 executing the same instruction set. P1 has a 3 GHz clock rate and a CPI of 1.5. P2 has a 2.5 GHz clock rate and a CPI of 1.0. P3 has a 4.0 GHz clock rate and has a CPI of 2.2. Which processor has the highest performance expressed in instructions per second? (1 GHz = $10^9$ Hz)

CPI = # clock cycles/# instructions

Clock rate = # clock cycles/second

Performance (# Instructions/ second) =

Clock rate/CPI =

(# clock cycles/# instructions) / (# clock cycles/second) =

# instructions / second

P1 performance = $3\times10^9/1.5 = 2\times10^9$

P2 performance = $2.5\times10^9/1 = 2.5\times10^9$ (highest performance)

P3 performance = $4\times10^9/2.2 = 1.8\times10^9$

# CSE 490/590 Spring 2023 Homework 1 Solution

3) a) Provide the assembly language instruction for the following R type instruction: [Use Green Sheet]

0000 0010 0001 0000 1000 0000 0010 $0000_2$

| 000000 | 10000 | 10000 | 10000 | 00000 | 100000 |
|---|---|---|---|---|---|
| Op=0 | rs=16 | rt=16 | rd=16 | shamt=0 | funct=32 |

add $s0,$s0,$s0

Green Sheet specifies the values for respective registers and opcode

b) Provide the binary of the following R type Instruction [Use Green Sheet]
sub $v1,$v1,$v0

| Op=0 | rs=3 | rt=2 | rd=3 | shamt=0 | funct=34 |
|---|---|---|---|---|---|
| 000000 | 00011 | 00010 | 00011 | 00000 | 100010 |

000000 00011 00010 00011 00000 $100010_2$

4) For the following C statement, what is the corresponding MIPS assembly code? Assume that the variables f, g, and h, are given and are assigned to registers $s2, $s3, and $s4 respectively. Use a minimal number of MIPS assembly instructions.

f = (g + h) - 5;

add $s2,$s3,$s4

addi $s2,$s2,-5

5)

a) Consider a byte-addressable memory system with the following contents:

| Memory Location | Value |
|---|---|
| 0x2000 | 01 |
| 0x2001 | 56 |
| 0x2002 | 70 |
| 0x2003 | 12 |
| 0x2004 | 23 |
| 0x2005 | 45 |
| 0x2006 | 67 |
| 0x2007 | 89 |

lw $s0,4($s1)

$s1 contains the address 0x2000. What will $s0 contain? [A big-endian system stores the most significant byte of a word at the smallest memory address and the least significant byte at the largest. A little-endian system, in contrast, stores the least-significant byte at the smallest address –wiki ]

Big endian representation => $s0 will hold the value 0x23456789

# CSE 490/590 Spring 2023 Homework 1 Solution

Little endian representation => $s0 will hold the value 0x89674523

b) Assume that $s0 contains the value 0x12121212 and $s1 contains the address 0x1FFF111.

Assume that the memory data, starting from address 0x1FFF111 is: 88 77 66 55. What will be the value of $s0 after the following code is executed:

lb $s0, 0($s1)

0xffffff88

The lb instruction sign-extends the byte into a 32-bit value, i.e. the most significant bit is copied into the upper 24 bits. (FYI: lbu is unsigned load byte instruction and so there is no sign extension; upper 24 bits will be zero in that case),

c) Assume that $t1 contains the value 0x0000000A. What will be the value of $t0 after the following code is executed:

sll $t0,$t1,5

0x0000000A => 0000 0000 0000 0000 0000 0000 0000 1010
0x0000000A << 5 => 0000 0000 0000 0000 0000 0001 0100 0000 = 0x00000140

6) Given the following instruction,
beq $s0, $s1, L1

replace it by using other instruction(s) that offers a much greater branching distance.

bne $s0, $s1, L2

j L1

L2:

By replacing branch instruction with a jump instructions, the address range allocated can be increased from $2^{16}$ to $2^{26}$ [Refer Green Sheet for the instruction format of J (jump) and I (branch) type instruction]. Hence the branching distance can be farther away in j.

7) Main routine M1 calls a procedure P1 (The return address is a value RM. Assume the initial stack pointer has a 32 bit value of 0x0700C. Procedure P1 will be using $s0, $s1, $s2 and need to be saved on the stack. RM (which is in $ra) need to be saved on the stack (since there is a call to another procedure). In P1, there is a call of procedure P2 (return address is a value RP1). Procedure P2 is likely to use $s4, $s5 (and hence need to be saved on stack). Show the SP value right after $s5 is saved. Also, show the stack contents (updated in this sequence).

| | |
|---|---|
| 0x0700C | |

<= $sp initial stack pointer position / Call P1

# CSE 490/590 Spring 2023 Homework 1 Solution

| | | |
|---|---|---|
| 0x07008 | $s0 | |
| 0x07004 | $s1 | |
| 0x07000 | $s2 | |
| 0x06FFC | $ra (RM) | <= $sp/ Call P2 |
| 0x06FF8 | $s4 | |
| 0x06FF4 | $s5 | <= $sp |
| 0x06FF0 | | |

The most recent value in $ra (RP1) need not be stored in the stack as it will not be replaced by any other return address since there are no more procedure call