CSE 490/590 Spring 2025 **Homework 1 Solution** **NOT GRADED**

Updated Content in Green and highlighted

[Question 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 MIPS-like instruction set and compute the overall CPI. **Show your work.**

Instruction Type	% of program	Cycles needed
Store and Load	40%	2
Branch	10%	4
Integer Add	30%	1
Shift	10%	1
Integer Multiply	10%	10

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

Explanation: 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.

[Question 2]

Consider three different processors P1, P2, and P3 executing the same instruction set.

Processor	Clock Speed in GHz	CPI
P1	3.0	1.5
P2	2.5	1.0
P3	4.0	2.2

Which processor has the highest performance expressed in instructions per second? Remember, 1GHz = 10^9Hz. **Show your work.**

CPI = # clock cycles / # instructions

Clock rate = # clock cycles / # seconds

Performance =

Clock rate / CPI =

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

~~(# clock cycles~~ / # seconds) / ~~(# clock cycles~~ / # instructions) =

instructions / # seconds

Thus:

P1 performance = 3×10^9 / 1.5 = 2×10^9

P2 performance = 2.5×10^9 / 1 = 2.5×10^9 (highest performance)

P3 performance = 4×10^9 / 2.2 = 1.81×10^9

[Question 3]

a. Provide the assembly language instruction for the following R type instruction which is written in binary: [Use the MIPS Green Sheet]

```
0000 0010 0001 0000 1000 1000 0010 0000
000000 10000 10000 10001 00000 100000
opcode    rs    rt    rd shamt  funct
     0    16    16    17     0     32(0x20 in hex)
R-type   $s0   $s0   $s1     0   0x20
Answer: add $s1, $s0, $s0
```

b. Provide the binary of the following R type Instruction [Use the MIPS Green Sheet]

```
sub $v1,$v1,$v0
     opcode    rs    rt    rd shamt  funct
          0   $v1   $v0   $v1     0   0x22
  Answer: 000000 00011 00010 00011 00000 100010
```

[Question 4]

For the following C statement, what is the corresponding MIPS assembly code? Assume that the variables `f`, `g`, `h`, and `i`, are given and are assigned to registers $s2, $s3, $s4, and $s5 respectively. Use a minimal number of MIPS assembly instructions.

```
f = (g + h) - (5 + i);
add $t0, $s3, $s4  // Computes (g + h)
addi $t1, $s5, 5   // Computes (5 + i)
sub $s2, $t0, $t1  // Computes (g + h) - (5 + i)
```

[Question 5]

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

Memory Location	Value
0x2000	0x01
0x2001	0x56
0x2002	0x70
0x2003	0x12
0x2004	0x23
0x2005	0x45
0x2006	0x67
0x2007	0x89

a. Assume `$s1` contains the value `0x2000`. After executing the instruction `lw $s0, 4($s1)` what will `$s0` contain? Use big-endian.

`$s0 contains: 0x23456789`

b. Assume `$s0` contains the value `0x12121212` and `$s1` contains the value `0x2006`. After executing the instruction `lb $s0, 1($s1)` what will `$s0` contain?

`$s0 contains: 0xFFFFFF89`

The `lb` instruction sign-extends the byte into a 32-bit value, i.e. the most significant bit of the loaded byte is copied into the upper 24 bits (this bit is a `1` since `0x89` = `0b10001001`). (FYI: the `lbu`, load byte unsigned, instruction does not do sign extension, the upper 24 bits will be zero in this case).

[Question 6]

Assume `$t1` contains the value `0x0000000A`. What will the value of `$t1 $t0` be after the following instruction is executed:

```
sll $t0, $t1, 5
```

`0x0000000A = 0b 0000 0000 0000 0000 0000 0000 0000 1010`

`$t1 = [0000 0000 0000 0000 0000 0000 0000 1010]`

`$t1 << 5 = 00000[0000 0000 0000 0000 0000 0001 010 ]`

`$t1 << 5 = [0000 0000 0000 0000 0000 0001 0100 0000]`

`$t0 = $t1 << 5 = 0x00000140`

`$t0 in binary: 0000 0000 0000 0000 0000 0001 0100 0000`

`$t0 in hexadecimal: 0x00000140`

[Question 7]

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 instruction, the target address range can be increased from 2^16 to 2^26 [Refer to the MIPS Green Sheet for the instruction format of J (jump) and I (branch) type instructions]. Hence the target location can be farther away in j.

[Question 8]

Main routine `M1` calls a procedure `P1` with the instruction `jal P1`. The return address in the main routine when it returns from the procedure `P1` is labelled as `RM`. The stack pointer has an initial value of `0x0700C`.

Procedure `P1` will use save registers `$s0` and `$s1`, and hence need to save these on the stack. Procedure `P1` calls another procedure `P2`, with a return address (in `P1`) labeled `RP1`.

Note, `RM` (which is stored in `$ra`) needs to be saved on the stack and restored eventually to `$ra` to return to the Main routine (since there is a call to another procedure).

Procedure `P2` uses save register `$s4`, (and hence needs to be saved on the stack). Show the stack pointer value right after `$s4` is saved. Also, show the stack contents (updated in the entire sequence).

Stack		
0x0700C		Initial $sp (stack pointer)
0x07008	$s0	
0x07004	$s1	
0x07000	$ra (RM)	$sp right after call to P2
0x06FFC	$s4	$sp right after $s4 is saved
0x06FF8		

The value of the stack pointer (`$sp`) after `$s4` is saved is `0x06FFC`.

The most recent value in `$ra` (`RP1`) does not need to be saved on the stack as it will not be replaced by any other return address since `P2` does not make any calls.