**Week 1 – Homework Problems**

Eric Vara

The University of Arizona Global Campus

CPT 301: Computer Organization & Architecture

Professor Austine Ohwobete

November 14, 2023

**Chapter 1  
1.1**

1. **Desktop Computers:** These are personal computers designed for regular use at a single location. They are larger and more powerful than portable computers, with separate components including a monitor, keyboard, and mouse.
2. **Laptops:** Also known as notebooks, these are portable computers that combine all the components of a desktop computer into a single device that is small enough to use on one's lap.
3. **Servers:** These are computers designed to provide data, services, or programs to other computers over a network. They often have powerful processors, lots of memory, and large storage capacities.
4. **Embedded Systems:** These are computers built into other devices, not typically thought of as computers. They are specialized for particular tasks and are found in a vast array of devices from digital watches and MP3 players to the control systems of airplanes.

**1.6**

P1 has a global CPI of 2.6 and needs 2,756,000 clock cycles to finish the program. Processor P2 has a global CPI of 2 and needs 2,120,000 clock cycles. So, Processor P2 is faster as it requires fewer clock cycles.

**1.11**

**1.12**

**Clock Rate and Performance Fallacy (1.12.1):**

* Processor P1 has a MIPS value of 4444.44.
* Processor P2 has a MIPS value of 4000.00.

This shows that the processor with the higher clock rate (P1) indeed has a higher MIPS, suggesting a better performance in this aspect.

**Number of Instructions Executed in the Same Time (1.12.2):**

* Processor P2 can execute 900,000,000 instructions in the same time it takes for P1 to execute 1.0E9 instructions.

**MIPS Fallacy (1.12.3):**

* Based on the calculation, Processor P1 has higher MIPS compared to P2, which could imply better performance. However, MIPS alone isn't a comprehensive measure of performance as it doesn't account for the type of instructions or other factors such as multi-threading or specific task optimizations.

**MFLOPS Figures (1.12.4):**

* Processor P1 has an MFLOPS value of 1777.78.
* Processor P2 has an MFLOPS value of 1600.00.

These values indicate that P1 performs better in terms of floating-point operations per second, which is consistent with having a higher clock rate and MIPS. However, like MIPS, MFLOPS should not be the sole metric for comparing overall performance

**Chapter 2**

**2.1**

To convert this into MIPS assembly code, assuming f, g, h, and i are 32-bit integers and using a minimal number of MIPS assembly instructions, the code would look something like this:

sub $t0, $h, 5 # t0 = h - 5

add $f, $g, $t0 # f = g + t0

* The sub instruction subtracts 5 from the contents of register $h (which holds the value of variable h) and stores the result in temporary register $t0.
* The add instruction adds the contents of register $g (which holds the value of variable g) to the contents of register $t0 and stores the result in register $f (which holds the value of variable f).

**2.2**

The first instruction add f, g, h adds the contents of registers g and h and stores the result in register f. The second instruction add f, i, f adds the contents of registers i and f and stores the result back into register f. So, the corresponding C statement that represents these two operations would be:

f = g + h;

f = i + f;

**2.3**

sub $t0, $s2, $s3

sll $t0, $t0, 2

add $t1, $s6, $t0

lw $t2, 0($t1)

sw $t2, 32($s7)

Here’s a small explaination of the MIPS instructions:

* sub is used to subtract j from i.
* sll (shift left logical) multiplies the result by 4 (equivalent to shifting left by 2 bits), because each integer is 4 bytes and we need to convert an array index to a byte offset.
* add adds the base address of A to the byte offset to get the address of A[i - j].
* lw (load word) loads the integer at that address into a temporary register.
* sw (store word) stores the value into the array B at the 8th index, which is a byte offset of 32 from the base address of B.​

**2.13**

To convert these MIPS instructions into the corresponding C statement:

1. sll $t0, $s0, 2 and add $t0, $s6, $t0 calculate the address of A[f].
2. sll $t1, $s1, 2 and add $t1, $s7, $t1 calculate the address of B[g].
3. lw $s0, 0($t0) loads the value of A[f] into $s0, which is the register for f.
4. addi $t2, $t0, 4 calculates the address of A[f + 1].
5. lw $t0, 0($t2) loads the value of A[f + 1] into $t0.
6. add $t0, $t0, $s0 adds f to A[f + 1] and stores the result in $t0.
7. sw $t0, 0($t1) stores the result into B[g].

The C statement that represents this sequence of operations would be:

B[g] = A[f + 1] + f;