ICS 233, Term 072

Computer Architecture & Assembly Language

HW# 5

Q.1. We wish to compare the performance of two different computers: M1 and M2. The following measurements have been made on these computers:

Program	Time on M1	Time on M2
1	2.0 seconds	1.5 seconds
2	5.0 seconds	10.0 seconds

Program	Instructions executed on M1	Instructions executed on M2
1	5×10^9	6×10^9

(i) Which computer is faster for each program, and how many times as fast is it?

(ii) Find the instruction execution rate (instructions per second) for each computer when running program 1.

(iii) The clock rates for M1 and M2 are 3 GHz and 5 GHz respectively. Find the CPI for program 1 on both machines.

(iv) Suppose that program 1 must be executed 1600 times each hour. Any remaining time should be used to run program 2. Which computer is faster for this workload? Performance is measured here by the throughput of program 2.

Q.2. Suppose you wish to run a program P with 7.5×10^9 instructions on a 5 GHz machine with a CPI of 1.2.

(i) What is the CPU execution time?

(ii) When you run program P, it takes 3 seconds of wall time to complete. What is the percentage of the CPU time program P received?

Q.3. Consider two different implementations, M1 and M2, of the same instruction set.
There are five classes of instructions (A, B, C, D and E) in the instruction set. M1 has a clock rate of 4 GHz and M2 has a clock rate of 6 GHz.

Class	CPI on M1	CPI on M2
A	1	2
B	2	2
C	3	2
D	4	4
E	3	4

(i) Assume that peak performance is defined as the fastest rate that a computer can execute any instruction sequence. What are the peak performances of M1 and M2 expressed in instructions per second?

(ii) If the number of instructions executed in a certain program is divided equally among the classes of instructions, except that for class A, which occurs twice as often as each of the others, how much faster is M2 than M1?

Q.4. Consider two different implementations, M1 and M2, of the same instruction set. There are three classes of instructions (A, B, and C) in the instruction set. M1 has a clock rate of 6 GHz and M2 has a clock rate of 3 GHz. The CPI for each instruction class on M1 and M2 is given in the following table:

Class	CPI on M1	CPI on M2	C1 Usage	C2 Usage	C3 Usage
A	2	1	40%	40%	60%
B	3	2	40%	20%	15%
C	5	2	20%	40%	25%

The above table also contains a summary of the usage of instruction classes generated by three different compilers: C1, C2, and C3. Assume that each compiler generates the same number of instructions for a given program.

(i) Using C1 compiler on both M1 and M2, how much faster is M1 than M2?

(ii) Using C2 compiler on both M1 and M2, how much faster is M2 than M1?

(iii) If you purchase M1, which compiler would you use?

(iv) If you purchase M2, which compiler would you use?

(v) Which computer and compiler combination give the best performance?

Q.5. A benchmark program runs for 100 seconds. We want to improve the speedup of the benchmark by a factor of 3. We enhance the floating-point hardware to make floating point instructions run 5 times faster. How much of the initial execution time would floating-point instructions have to account for to show an overall speedup of 3 on this benchmark?

Q.6. Consider the following fragment of MIPS code. Assume that **a** and **b** are arrays of words and the base address of **a** is in **$a0** and the base address of **b** is in **$a1**. How many instructions are executed during the running of this code? If ALU instructions (**addu** and **addiu**) take 1 cycle to execute, load/store (**lw** and **sw)** take 5 cycles to execute, and the branch (**bne**) instruction takes 3 cycles to execute, how many cycles are needed to execute the following code (all iterations). What is the average CPI?

```
          addu $t0, $zero, $zero   # i = 0
          addu $t1, $a0, $zero     # $t1 = address of a[i]
          addu $t2, $a1, $zero     # $t2 = address of b[i]
          addiu $t3, $zero, 101    # $t3 = 101 (max i)
loop:     lw $t4, 0($t2)           # $t4 = b[i]
          addu $t5, $t4, $s0       # $t5 = b[i] + c
          sw $t5, 0($t1)           # a[i] = b[i] + c
          addiu $t0, $t0, 1        # i++
          addiu $t1, $t1, 4        # address of next a[i]
          addiu $t2, $t2, 4        # address of next b[i]
          bne $t0, $t3, loop       # loop if (i != 101)
```

Q.7. You are required to design a combinational logic block to be used for implementing the SLL instruction for shifting an operand left by a maximum of 31 bits. Assume that the shifter will get the operand to be shifted along with the 5 bits that determine the shift amount. Design the left shifter to minimize its area and delay. Hint: use 4x1 MUXs and 2x1 MUX to implement the left shifter.