Computer Architecture and Programming Languages Course: CO20-320241
Jacobs University Bremen November 11, 2019
Jovan Shandro

Homework 8

Problem 8.1

Solution:
i) We know 32 is 2^{-5}, so the first thing we do is find the binary representation of 25. It goes as follows:

$$25 \xrightarrow{/2} 12(1) \xrightarrow{/2} 6(0) \xrightarrow{/2} 3(0) \xrightarrow{/2} 1(1) \xrightarrow{/2} 0(1)$$

So 25 is represented as 11001_2. Thus $25/32 = 11001 \cdot 2^{-5} = 1.1001 \cdot 2^{-1}$. Adding 127 to the exponent we get 126 which in binary is represented as 1111110_2 as it is only 2 less than $2^7 = 128$. Since the number is positive, the sign bit will be zero, and the part after the decimal point above will be stored in the mantissa. So the IEEE754 single precision binary representation will be :

Sign Bit	Exponent	Mantissa
0	01111110	10010000000000000000000

ii) We first find the binary representations of 27 and 0.3515625 and then use them to find the IEEE754 single precision binary representation. It goes as follows:

$$27 \xrightarrow{/2} 13(1) \xrightarrow{/2} 6(1) \xrightarrow{/2} 3(0) \xrightarrow{/2} 1(1) \xrightarrow{/2} 0(1)$$

So 27 is represented as 11011_2. For 0.3515625:

$0.3515625 \cdot 2 = 0.703125 \quad (0)$
$0.7031250 \cdot 2 = 1.40625 \quad (1)$
$0.4062500 \cdot 2 = 0.8125 \quad (0)$
$0.8125 \cdot 2 = 1.625 \quad (1)$
$0.6250 \cdot 2 = 1.25 \quad (1)$
$0.25 \cdot 2 = 0.5 \quad (0)$
$0.5 \cdot 2 = 1 \quad (1)$

So our number is represented $11011.0101101_2 = 1.10110101101_2 \cdot 2^4$. So under the exponent $127 + 4 = 131$ will be stored. Since 131 is only 3 more than 128 its binary representation is 10000011. Since the number is again positive, the sign bit will be zero, and the part after the decimal point above will be stored in the mantissa. So the IEEE754 single precision binary representation will be :

Sign Bit	Exponent	Mantissa
0	10000011	10110101101000000000000

Problem 8.2

Solution:
Out of all five statements only the first one is true, all the others are false.

Problem 8.3

Solution:
So we have been given the instruction:

op	rs	rt	rd	shamt	funct
000000	10000	10101	01011	00000	100000
0	16	21	11	0	32

Since the opcode is 0, shift amount is 0, and the function code is 32, we have an addition instruction. The 2 sources are the registers 16 and 21, so $\$s0$ and $\$s5$ respectively, and the destination register is $\$t3$ (11). So the mips instruction is:

add $t3 $s0 $s5

Problem 8.4

Solution:
a) Knowing that MIPS instructions are 32 bit long, and that 6 bits are needed to store the opcode of the instruction j, we are left with 26 bits that can be used for the destination address.

b) One possible way would be to use the jr instruction. It jumps into the value stored in the specified register, and since registers store 32 bits, this instruction allows us to jump 'anywhere'.

Problem 8.5

Solution:
From the slides we know that CPU_Time = CPU_ClockCycles/Clock_Rate, and we also know that CPU_ClockCycles = Effective_CPI * Total_IC = Total_IC * $\sum_{k=1}^{n} IC_k \cdot CPI_k/Total\_IC$.

So, using also the fact that frequency is the respective IC over the total one ($IC_k/Total_{IC}$), for the given computers we have:

$Time_{P_1} = Total_{IC} * (1\cdot 0.6 + 2\cdot 0.1 + 3\cdot 0.2 + 4\cdot 0.1)/4GHz = 0.45\cdot 10^{-9} * Total_{IC}$.

In a similar way we calculate:

$Time_{P_2} = Total_{IC} * (2\cdot 0.8 + 4\cdot 0.2)/6GHz = 0.4\cdot 10^{-9} * Total_{IC}$.

https://www.overleaf.com/project/5dc1959129200c00014f22cb And since $(Time_{P_1} - Time_{P_2})/Time_{P_1} = (4.5 - 4)/4 = 0.125 \longrightarrow$ so P_2 is 12.5% times faster than P_1.

Problem 8.6

Solution:
We first find the frequence of each of the classes. Since all classes apart from A have same frequency, let it be α. So A has a frequency of $2\cdot\alpha$. Since all frequences add up to 1, we have that $2\cdot\alpha + \alpha\cdot(1+1+1+1) = 6\cdot\alpha = 1 \longrightarrow \alpha = 1/6$. Now we can calculate the CPU times as we did in the previous question:

$Time_{P_1} = Total_{IC} * (1\cdot(1/3) + (1/6)\cdot(3+3+4+2))/2GHz = (7/6)\cdot 10^{-9} * Total_{IC}$.

In a similar way we calculate:

$Time_{P_2} = Total_{IC} * (2\cdot(1/3) + (1/6)\cdot(3+2+3+3))/4GHz = 1.25\cdot 10^{-9} * Total_{IC}$.

We see $Time_{P_2}$ is less than $Time_{P_1}$, so P_2 is faster. And since $(Time_{P_1} - Time_{P_2})/Time_{P_1} = (28 - 15)/15 \approxeq 0.87 \longrightarrow$ so P_2 is 87% times faster than P_1.