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

Homework 8

Problem 8.1

Solution:

i) $\frac{25}{32} = 25 \cdot 32^{-1} = 25 \cdot 2^{-5}$

Now we perform conversion to binary:
$25/2 = 12 \to \mathbf{1}; \quad 12/2 = 6 \to \mathbf{0}; \quad 6/2 = 3 \to \mathbf{0}; \quad 3/2 = 1 \to \mathbf{1}; \ 1/2 = 0 \to \mathbf{1}$

Therefore, we have:
$25_{10} = 11001_2 = 1.1001 \cdot 2^{-4} \to \frac{25}{32} = 25 \cdot 2^{-5} = 1.1001 \cdot 2^{-4} \cdot 2^{-5} = 1.1001 \cdot 2^{-1}$

So, for the IEEE 754 single precision binary representation, we have:
* sign bit = 0, since we have a positive fraction;
* exponent is -1, so adding 127 we get $127 - 1 = 126 = 01111110_2$ (8 bits in total);
* mantissa will be the floating part, which is 1001000...0, until we have 23 bits.

The final representation would be: $0 \quad 01111110 \quad 10010000000000000000000$

ii) 27.3515625

First we convert 27 to binary:
$27/2 = 13 \to \mathbf{1}; \quad 13/2 = 6 \to \mathbf{1}; \quad 6/2 = 3 \to \mathbf{0}; \quad 3/2 = 1 \to \mathbf{1}; \ 1/2 = 0 \to \mathbf{1}$
$\to 27_{10} = 11011_2$

Next, we convert the floating part to binary:
$0.3515625 \cdot 2 = 0.703125 \to \mathbf{0}; \quad 0.703125 \cdot 2 = 1.40625 \to \mathbf{1}; \quad 0.40625 \cdot 2 = 0.8125 \to \mathbf{0}; \quad 0.8125 \cdot 2 = 1.625 \to \mathbf{1}; \quad 0.625 \cdot 2 = 1.25 \to \mathbf{1}; \quad 0.25 \cdot 2 = 0.5 \to \mathbf{0}; \quad 0.5 \cdot 2 = 1 \to \mathbf{1}$, and the rest is zero, so final representation is:
0.3515625 = 0.010110100...

Now, we find the scientific notation of the whole number 27.3515625 in binary:
11011.010110100.. = 1.1011010110100.. $\cdot 2^4$

As a result, we have:
* sign bit = 0 (positive number);
* exponent = 4, so adding 127 we get $127 + 4 = 131 = 10000011_2$S;
* mantissa will be: 1011010110100...0, until we have 23 bits.

Final representation: $0 \quad 10000011 \quad 10110101101000000000000$

Problem 8.2

Solution:

MIPS has an alignment restriction, that means words must start at addresses that are multiples of 4. $\to \boldsymbol{T}$
All MIPS instructions are 30 bits long. $\to \boldsymbol{F}$
MIPS can perform arithmetic operations on memory locations. $\to \boldsymbol{F}$
Parameters to functions are always passed via the stack. $\to \boldsymbol{F}$
A procedure jumps to the address stored in the stack pointer register after it finishes execution. $\to \boldsymbol{F}$

Problem 8.3
Solution:

We have `op` = 000000, which means that we have either `add` or `sub`, and since `func` = 100000 = 32_{10}, we have the operation `add`. The registers are: register `rs` = 10000 = $16_{10} \rightarrow$ `$s0`, register `rt` = 10101 = $21_{10} \rightarrow$ `$s5`, and `rd` = 01011 = $11_{10} \rightarrow$ `$t3`. So, the MIPS assembler code would be: `add $t3, $s0, $s5`

Problem 8.4
Solution:

a) Since for the `op` code we need only only 6 bits and `j` needs only one argument after (destination address), we find that for the destination address can be used 32 - 6 = 26 bits.

b) Since in the previous case, using `j`, more than 26 bits destinations cannot be covered, we can use `jr` (jump to register), so that we jump to the value stored in the register, which can be up to 32 bits.

Problem 8.5
Solution:

Class	CPI on P1	Freq*CPI$_1$	CPI on P2	Freq*CPI$_2$	Frequency
A	1	0.6	2	1.2	60%
B	2	0.2	2	0.2	10%
C	3	0.3	2	0.2	10%
D	4	0.4	4	0.4	10%
E	3	0.3	4	0.4	10%
		$\sum$ = 1.8		$\sum$ = 2.4	

$$\text{CPU time} = \frac{\text{Instruction count * CPI}}{\text{Clock rate}}$$

So, we have:

$$\text{CPU time for P1} = \frac{\text{IC * CPI}_1}{\text{Clock rate of P1}} = \frac{\text{IC}\cdot 1.8}{4\cdot 10^9} = 0.45 \cdot 10^{-9} \cdot \text{IC}$$

$$\text{CPU time for P2} = \frac{\text{IC * CPI}_2}{\text{Clock rate of P1}} = \frac{\text{IC}\cdot 2.4}{6\cdot 10^9} = 0.4 \cdot 10^{-9} \cdot \text{IC}$$

As we can observe from the results (0.4 < 0.45), P2 will finish rendering the image first. The exact ratio would be:

$\frac{\text{CPU time for P1}}{\text{CPU time for P2}} = \frac{0.45}{0.4} = 1.125$, which means that P2 is 12,5% faster than P1.

Problem 8.6
Solution:

Considering the fact that we have 5 instruction classes and we want to divide equally among the classes except for class A, which will occur twice as often as the others, it's the same as if we consider having 6 classes, and A counts for 2 of them. So, with 6 classes of same frequency, we get that the frequency of each class will be 1/6, except for A, which will be 2/6 (as percentages they'll be 16.666... and 33.333...). Therefore, we construct the following table:

Class	CPI on P1	Freq*CPI$_1$	CPI on P2	Freq*CPI$_2$	Frequency
A	1	1/3	2	2/3	2/6
B	3	1/2	3	1/2	1/6
C	3	1/2	2	1/3	1/6
D	4	2/3	3	1/2	1/6
E	2	1/3	3	1/2	1/6
		$\sum$ = 7/3		$\sum$ = 5/2	

CPU time = $\frac{\text{Instruction count * CPI}}{\text{Clock rate}}$

So, we have:

CPU time for P1 = $\frac{\text{IC * CPI}_1}{\text{Clock rate of P1}} = \frac{\text{IC}\cdot\frac{7}{3}}{2\cdot 10^9} = \frac{7}{6}\cdot 10^{-9}\cdot \text{IC}$

CPU time for P2 = $\frac{\text{IC * CPI}_2}{\text{Clock rate of P1}} = \frac{\text{IC}\cdot\frac{5}{2}}{4\cdot 10^9} = \frac{5}{8}\cdot 10^{-9}\cdot \text{IC}$

Now we find the ratio:

$\frac{\text{CPU time for P1}}{\text{CPU time for P2}} = \frac{7/6}{5/8} = \frac{28}{15} =$1.8666..., so P2 is $\approx$ 86,7% faster than P1.