Computer Architecture and Programming Languages
Jacobs University Bremen
Fjolla Dedaj

Course: CO20-320241
11th of November, 2019

Homework 8

Problem 8.1

Compute the IEEE 754 single precision binary representation of the following numbers:

(i) $\frac{25}{32}$

(ii) 27.3515625

Solution:

(i) $\frac{25}{32} = \frac{25}{2^5} = \frac{11001}{2^5} = 0.11001 = 1.1001 \times 2^{-1}$

Exponent = 127 + (-1) = 126 = 1111110
Sign = 0 (positive number)
Fraction = 10010000000000000000000

S	E(exponent)	F(fraction)
0	01111110	10010000000000000000000

(ii) 27.3515625

$27 = 11011_2$

0.3515625×2	0.703125	0
0.703125×2	1.40625	1
0.40625×2	0.8125	0
0.8125×2	1.625	1
0.625×2	1.25	1
0.25×2	0.5	0
0.5×2	1	1

$27.3515625 = 11011.01011010...0 = 1.1011010110100...0 \times 2^4$
$E = 127 + 4 = 131_{10} = 10000011_2$

S	E(exponent)	F(fraction)
0	10000011	10110101101000000000000

Problem 8.2

Solution:

☑ = True

☒ = False

- MIPS properties:

  ☑ MIPS has an alignment restriction, that means words must start at addresses that are multiples of 4.

  ☒ All MIPS instructions are 30 bits long.

  ☒ MIPS can perform arithmetic operations on memory locations.

  ☒ Parameters to functions are always passed via the stack.

  ☒ A procedure jumps to the address stored in the stack pointer register after it finishes execution.

Problem 8.3

In MIPS assembly language, registers \$s0 to \$s7 map onto register 16 to 23, and registers \$t0 to \$t7 map onto register 8 to 15. The opcode for addition and subtraction is 0. The function code is 32 for addition and 34 for subtraction. Given all this, translate the following binary word into a MIPS instruction:

000000 10000 10101 01011 00000 100000

Solution:

Op code	Source 1	Source 2	Destination	Shift Amount	Fraction Code
000000	10000	10101	01011	00000	100000

$$\Downarrow$$

0	16	21	11	0	32
add/subtract	\$s0	\$s5	\$t3	No shift	add

Thus, the instruction is:

add \$t3, \$s0, \$s5

Problem 8.4

a) How many bits can be used for the destination address in the j (jump) instruction?

Solution:
There is room for a 26-bit address. The 26-bit target address field is transformed into a 32-bit address. This is done at run-time, as the jump instruction is executed.

b) If the address representation within the instruction does not cover the full 32-bit range (as in the previous question), what can be done to still be able to jump anywhere (assuming 32-bit addresses)?

Solution:
As mentioned earlier, the 26-bit target address field can be transformed into a 32-bit address. This is done at run-time, as the jump instruction is executed. jr instruction has this behavior.

Problem 8.5

Solution:

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

CPU time for P1= $\frac{\text{Instruction Count x CPI}}{\text{Clock Rate}} = \frac{13 \times 1.8}{4} = 5.85$

CPU time for P2= $\frac{\text{Instruction Count x CPI}}{\text{Clock Rate}} = \frac{14 \times 2.4}{6} = 5.6$

Therefore, $5.85/5.6 \approx 1.045$ meaning that P2 is 4.5% faster and will render the image first.

Problem 8.6

Solution:

Class	CPI on P1	CPI on P2
A	1	2
B	3	3
C	3	2
D	4	3
E	2	3

T(P1)/T(P2) =(1*2+3+3+4+2)4/(2*2+3+2+3+3)2 = 14*4/15*2 = 28/15 $\approx$ 1.866

Instruction count is not included because it is the same, therefore the ratio doesn't change.

Therefore, P2 is 86.6% faster.