Computer Architecture and Programming Languages
Jacobs University Bremen
Fjolla Dedaj

Course: CO20-320241
14th of October, 2019

Homework 5

Problem 5.1

Solution:

a) $14 + 37 = 1110_2 + 100101_2 = 110011$

14_{10} :

$$\left.\begin{array}{r|l} 2\lfloor\underline{14} & 0 \\ 2\lfloor\underline{7} & 1 \\ 2\lfloor\underline{3} & 1 \\ 2\lfloor\underline{1} & 1 \end{array}\right\} = 1110_2$$

37_{10} :

$$\left.\begin{array}{r|l} 2\lfloor\underline{37} & 1 \\ 2\lfloor\underline{18} & 0 \\ 2\lfloor\underline{9} & 1 \\ 2\lfloor\underline{4} & 0 \\ 2\lfloor\underline{2} & 0 \\ 2\lfloor\underline{1} & 1 \end{array}\right\} = 100101_2$$

$$\begin{array}{r} {\scriptstyle 1\,1} \\ 001110 \\ +100101 \\ \hline 110011 \end{array}$$

b) $12 - 27 = 1110_2 + 100101_2 = 110001$

12_{10} :

$$\left.\begin{array}{r|l} 2\lfloor\underline{12} & 0 \\ 2\lfloor\underline{6} & 0 \\ 2\lfloor\underline{3} & 1 \\ 2\lfloor\underline{1} & 1 \end{array}\right\} = 1100_2$$

27_{10} :

$$\left.\begin{array}{r|l} 2\lfloor\underline{27} & 1 \\ 2\lfloor\underline{13} & 1 \\ 2\lfloor\underline{6} & 0 \\ 2\lfloor\underline{3} & 1 \\ 2\lfloor\underline{1} & 1 \end{array}\right\} = 11011_2$$

Flip the bits and add 1 to get -27 (we need 6 bits in total to represent the sign also): $!(11011) + 1 = 00100 + 1 = 100101$

$$\begin{array}{r} {\scriptstyle 1\,1} \\ 001100 \\ +100101 \\ \hline 110001 \end{array}$$

c) $69 + 58 = 0110\ 1001 + 0101\ 0100 = 1\ 0010\ 0111$

$$\begin{array}{rl} 69 = 0110\ 1001 & \\ 58 = 0101\ 1000 & + \\ \hline 1100\ 0001 & \\ 0110\ 0000 & + \\ \hline 1\ 1010\ 0001 & \end{array}$$

d) $275 + 642 = 100100010111$

$$\begin{array}{rl} 275 = 0010\ 0111\ 0101 & \\ 642 = 0110\ 0100\ 0010 & + \\ \hline 1000\ 1011\ 0111 & \\ 0000\ 0110\ 0000 & + \\ \hline 1001\ 0001\ 0111 & \end{array}$$

e) $6AF + 23C = 8EB$

```
   1
  6AF         F = 15          carry: 1
+ 23C         C = 12           A = 10
  8EB         ------            +  3
                27            -------
              - 16  +              E
              ------
              11 => B
              + 1 carry
```

f) $594 - 3A8 = 1EC$

```
   24
 4 8 20
  594
-  3A8
  1EC
```

Problem 5.2

Solution:

MIPS Code:

```
a) a = b + c
add $t0, $s0, $s1
```

name of operation, destination, 1st source, 2nd source

```
b) a = b - d + c
sub $t0, $s0, $s2
add $t0, $t0, $s1
```

First subtract d from b and store it in $t0.
Then, add c to the resultand store it in $t0 also.

```
c) a = 3 * b
add $t0, $s0, $s0
add $t0, $t0, $s0
```

First add b and b and store the result in $t0.
Then, add another b to the result.

```
d) q = (1+b) * 2
add $t0, 1, $s0
add $t0, $t0, $t0
```

First add 1 and b and store the result in a. Then, add the result to itself since we have 2 times the same thing

Problem 5.3

Solution:

```
a) a = b + c
add $t0, $s0, $s1

b) a = b - d + c
sub $t0, $s0, $s2
add $t0, $t0, $s1
```

	Op	Rs	Rt	Rd	Shamt	Adr/funct
a)	**000000**	10000	10001	01000	00000	100000
b)	**000000**	10000	10010	01000	00000	100010
b)	**000000**	01000	10001	01000	00000	100000

Problem 5.4
Solution:

MIPS Code:

```
B[5] = A[4] + A[2]
lw $t0, 16($s0)
lw $t1, 8($s0)
add 20($s1), $t1, $t0
```

Calculate the offset: A[i] = 4 * i. So:

A[4] = 4 * 4 = 16

A[2] = 4 * 2 = 8

Add the result in B[5] where the offset is: B[5] = 4 * 5 = 20

Problem 5.5
Solution:

MIPS Code:

```
add $t1, $t0, 2  ——> (x + 2)
add $t1, $t1, $t1 ——> (x + 2) + (x + 2)
add $t1, $t1, $t1 ——> [(x + 2) + (x + 2) + (x + 2) + (x + 2)]
lw $t1, $t1($s0)
add $t2, $t0, 7  ——> (x + 7)
add $t2, $t2, $t2 ——> (x + 7) + (x + 7)
add $t2, $t2, $t2 ——> [(x + 7) + (x + 7) + (x + 7) + (x + 7)]
lw $t2, $t2($s0)
add $t3, $t0, $t0 ——> x + x
add $t3, $t3, $t3 ——> (x + x) + (x + x)
lw $t3, $t3($s1)
add $t4, $t1, $t2 ——> A[x + 2] + A[x + 7]
sw $t4, $t3($s1) ——> store the value in a temporary register
```

Problem 5.6
Solution:

The register instruction format for the instruction addi uses 6 bits for the opcode, 5 bits for the first register, 5 bits for the second register, and 16 bits for the constant value that is to be added. Assume that a detailed analysis of machine code has revealed that not more than 12 registers are needed for the processor and therefore it has been decided to reduce the number of general purpose registers to 16. How and why could the instruction format for the addi instruction be changed?

Addi copies the most significant bit (which is the sign bit) to all upper 16 bits of the destination. This is called sign extension. A negative operand would propagate 1's to the upper bits and watches for automatic sign extensions in arithmetic operations in MIPS. 16 higher bits of immediate operand are zeros, therefore it does the job.

To solve this assignment sheet, I collaborated with Dion Dermaku.