*Advanced Computer Architecutre Lab 2018*

1 – Warm-up questions

* 1. ADD R2 R0 5
  2. SUB R2 R0 5
  3. SUB R2 R2 R3
  4. Regarding “>” and “>=”, it can be implemented by replacing the src0 and src1 numbers in JLT and JLE instructions.
  5. In order to load a 32 bit constant into a register, one can use multiple instructions. For example, if the constant is 0x89ABCDEF and the register is R2:  
     OR R2 R0 0XCDEF

LHI R2 0X89AB

1. – Example program
   1. The program does the first step in an array scan algorithm.

Namely, it makes an addition of each couple of memory content and stores it the first.

Memory contents before the run:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Memory contents after the run:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
| 1 | 3 | 5 | 7 | 9 | 11 | 13 | 7 |

* 1. The inputs are stored in memory addresses 15-22 (including).
  2. The outputs are stored in memory addresses 15-21 (including).
  3. The commented version of the assembly program:

asm\_cmd(ADD, 2, 1, 0, 15); // 0: R2 = 15

asm\_cmd(ADD, 3, 1, 0, 1); // 1: R3 = 1

asm\_cmd(ADD, 4, 1, 0, 8); // 2: R4 = 8

asm\_cmd(JEQ, 0, 3, 4, 11); // 3: if (R3 == R4) goto 11

asm\_cmd(LD, 5, 0, 2, 0); // 4: R5 = MEM[R2]

asm\_cmd(ADD, 2, 2, 1, 1); // 5: R2++

asm\_cmd(LD, 6, 0, 2, 0); // 6: R6 = MEM[R2]

asm\_cmd(ADD, 6, 6, 5, 0); // 7: R6 += R5

asm\_cmd(ST, 0, 6, 2, 0); // 8: MEM[R2] = R6

asm\_cmd(ADD, 3, 3, 1, 1); // 9: R3++

asm\_cmd(JEQ, 0, 0, 0, 3); // 10: goto 3

asm\_cmd(HLT, 0, 0, 0, 0); // 11: HALT

* 1. The program can use immidieates of the numbers 0-7 instead of using the memory content, and only stores the outputs. The new program:

asm\_cmd(ADD, 2, 0, 0, 15); // 0: R2 = 0

asm\_cmd(ADD, 3, 1, 0, 1); // 1: R3 = 1

asm\_cmd(ADD, 4, 1, 0, 8); // 2: R4 = 8

asm\_cmd(JEQ, 0, 3, 4, 11); // 3: if (R3 == R4) goto 11

asm\_cmd(ADD, 5, 0, 2, 0); // 4: R5 = R2

asm\_cmd(ADD, 2, 2, 1, 1); // 5: R2++

asm\_cmd(ADD, 6, 0, 2, 0); // 6: R6 = R2

asm\_cmd(ADD, 6, 6, 5, 0); // 7: R6 += R5

asm\_cmd(ST, 0, 6, 2, 0); // 8: MEM[R2] = R6

asm\_cmd(ADD, 3, 3, 1, 1); // 9: R3++

asm\_cmd(JEQ, 0, 0, 0, 3); // 10: goto 3

asm\_cmd(HLT, 0, 0, 0, 0); // 11: HALT