Introduction to Computing System

Homework 4 Answer

Due date: 00:00 on July 17, 2018
Yale N. Patt
Jingwen Leng
Summer 2018

Homework

6.7

This program adds together the corresponding elements of two lists of numbers (vector addition). One list starts at address x300e and the other starts at address x3013. The program finds the length of the two lists at memory location x3018. The first element of the first list is added to the first element of the second list and the result is stored back to the first element of the first list; the second element of the first list is added to the second element of the second list and the result is stored back to the second element of the first list; and so on.

6.11

This program increments each number in the list of numbers starting at address A and ending at address B. The program tells when it's done by comparing the last address it loaded data from with the address B. When the two addresses are equal, the program stops incrementing data values.

```
0x3000 0010 000 011111111        ( LD R0, x3100 )
0x3001 0010 001 011111111        ( LD R1, x3101 )
0x3002 0001 001 001 1 00001      ( ADD R1, R1, #1 )
0x3003 1001 001 001 111111       ( NOT R1, R1 )
0x3004 0001 001 001 1 00001      ( ADD R1, R1 #1 )
0x3005 0001 011 000 0 00 001     ( l ADD R3, R0, R1 )
0x3006 0000 010 000000101        ( BRz Done )
0x3007 0110 010 000 000000       ( LDR R2, R0, #0 )
0x3008 0001 010010100001         ( ADD R2, R2, #1 )
0x3009 0111 010000000000         ( STR R2, R0, #0 )
0x300a 0001 000000100001         ( ADD R0, R0, #1 )
0x300b 0000 111111111001         ( BRnzp l )
0x300c 1111 000000100101         ( Done HALT )
```

6.14

```
0x3000 0101 010 010 1 00000      ( AND R2, R2, #0 )
0x3001 0001 001 001 1 11111      ( ADD R1, R1, #-1 )
0x3002 0001 001 001 1 11111      ( ADD R1, R1, #-1 )
0x3003 0001 001 001 1 11111      ( ADD R1, R1, #-1 )
0x3004 0000 100 000000010        ( BRn x3007 )
0x3005 0001 010 010 1 00001      ( ADD R2, R2, #1 )
0x3006 0000 111 111111010        ( BRnzp x3001 )
0x3007 1111 0000 00100101        ( TRAP 0x25 )
```

The possible values for R1 are: 9, 10, and 11.

6.18

```
0x3000 0101 0000 0010 0000      ( ADD R0, R0, #0 )
0x3001 1010 0010 0000 1011      ( LDI R1, PDIVIDEND )
0x3002 1010 0100 0000 1011      ( LDI r2, PDIVISOR )
0x3003 1001 0110 1011 1111      ( NOT R3, R2 )
0x3004 0001 0110 1110 0001      ( ADD R3, R3, #1 )
0x3005 0001 0010 0100 0011      ( LOOP ADD R1, R1, R3 )
0x3006 0000 1000 0000 0010      ( BRn REMN )
0x3007 0001 0000 0010 0001      ( ADD R0, R0, #1 )
0x3008 0000 1111 1111 1100      ( BRnzp LOOP )
0x3009 0001 0010 0100 0010      ( REMN ADD R1, R1, R2
0x300a 1011 0000 0000 0100      ( STI R0, PQUO )
```

7.4

Symbol	Address
Test	x301F
Finish	x3027
Save3	x3029
Save2	x302A

7.9

The .END pseudo-op tells the assembler where the program ends.

Any string that occurs after that will be disregarded and not processed by the assembler.

It is different from HALT instruction in very fundamental aspects:

1. It is not an instruction, it can never be executed.
2. Therefore it does not stop the machine.
3. It is just a marker that helps the assembler to know where to stop assembling.

7.16
This program counts the number of even and the number of odd integers.
It stores the number of even integers in R3 and stores the number of odd integers in R4.

7.18

```
a) LDR   R3,R1,#0
b) NOT   R3,R3
c) ADD   R3,R3,#1
or
a) LDR   R3,R1,#0
b) NOT   R4,R4
c) ADD   R4,R4,#1
```

7.21
This program counts the number of negative values in memory locations 0x4000 - 0x4009 and stores the result in memory location 0x5000.

```
0101 0000 0010 0000 ( AND R0, R0, #0 )
0001 0100 0010 1010 ( ADD R2, R0, #10 )
0010 0010 0000 1010 ( LD R1, MASK )
0010 0110 0000 1010 ( LD R3, PTR1 )
0110 1000 1100 0000 ( LDR R4, R3, #0 )
0101 1001 0000 0001 ( AND R4, R4, R1 )
0000 0100 0000 0001 ( BRz NEXT )
0001 0000 0010 0001 ( ADD R0, R0, #1 )
0001 0110 1110 0001 ( ADD R3, R3, #1 )
0001 0100 1011 1111 ( ADD R2, R2, #-1 )
0000 0011 1111 1001 ( BRp LOOP )
1011 0000 0000 0011 ( STI R0, PTR2 )
1111 0000 0010 0101 ( HALT )
1000 0000 0000 0000
0100 0000 0000 0000
0101 0000 0000 0000
```

7.25
This is an assembler error. The number 0xFF004 does not fit in one LC-3 memory location and therefore this .FILL cannot be assembled.

9.16

Error 1: The line VALUE .FILL X30000 will generate an assembly error because 0x30000 does not fit in one LC-3 memory location.