CISC 260 Machine Organization and Assembly Language

Assignment # 3 Solution

1. For this part, we will assume these addresses as registers from R0 through R7:

000 : R0	001 : R1	010 : R2	011 : R3
100 : R4	101 : R5	110 : R6	111 : R7

Dissembled code:

Addr	Code	15:14	13:10	9	8:6	5:3	2:0	opcode	operation	explanation
0x00	07FF	00	0001	1	111	111	111	SUB	SUB R7, R7, R7	R7 = R7 – R7
0x02 ***L₀***	244E	00	1001	0	001	001	110	BRZ	BRZ R1, (0x0E)	If value at R1 = 0, branch to ***END***
0x04	1E54	00	0111	1	001	010	100	AND	AND R4, R1, R2	R4 = R1 & R2
0x06	250A	00	1001	0	100	001	010	BRZ	BRZ R4, (0x0A)	If value at R4 = 0, branch to ***L₁***
0x08	23C7	00	1000	1	111	000	111	INCR	INCR R7, R7	R7 = R7 + 1
0x0A ***L₁***	0E41	00	0011	1	001	000	001	SRL	SRL R1, R1	R1 = R1 >> 1
0x0C	24C2	00	1001	0	011	000	010	BRZ	BRZ R3, (0x02)	If value at R3 = 0, branch to ***L₀***
0x0E ***END***	3C00	00	1111	0	000	000	000	HALT	HALT	Stop

C code for better understanding:

```
#include <stdio.h>
int main()
{
    int R7 = 0;
    int R1 = 53;
    int R2 = 1;
    int R4;
    int c = 1;
    while(R1 != 0){
        printf("Iteration: %d \n", c++);
        R4 = R1 & R2;
        printf("R4: %d \n", R4);
        if( R4) {
            R7++;
            printf("R7 updated: %d \n", R7);
        }
        R1 = R1 >> 1;
        printf("R1 rshifted: %d \n \n", R1);
    }
    printf("R1: %d \nR2: %d \nR4: %d\nR7: %d", R1, R2, R4, R7);
}
```

2. Values of the 8 registers after the execution of the program in part 1: (register values at the end of each iteration/cycle are shown)

	Initial	1	2	3	4	5	6	7
R0	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
R1	0011 0101	0011 0101	0001 1010	0000 1101	0000 0110	0000 0011	0000 0001	0000 0000
R2	0000 0001	0000 0001	0000 0001	0000 0001	0000 0001	0000 0001	0000 0001	0000 0001
R3	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
R4	0000 0000	0000 0001	0000 0001	0000 0000	0000 0001	0000 0000	0000 0001	0000 0001
R5	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
R6	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
R7	0001 0000	0001 0000	0000 0001	0000 0001	0000 0010	0000 0010	0000 0011	0000 0100

3. Translated **assembly code**:

```
c = 0;
while (1) {
        if (a > b) {
              a = a - b;
              c++;
        } else {
              break;
        }
}
return c;
```

0x00	SUB R7, R7, R7	0000 0111 1111 1111	07FF
0x02	SUB R4, R0, R1	0000 0110 0000 1100	070C
0x04	BREZ R4, (0x0A)	0010 1001 0000 1010	290A
0x06	SUB R0, R0, R1	0000 0110 0000 1000	0608
0x08	INCR R7, R7	0010 0011 1100 0111	23C7
0x0A	JUMP (0x0E)	0011 1000 0000 1110	380E
0x0C	JUMP (0x02)	0011 1000 0000 0010	3802
0x0E	HALT	0011 1100 0000 0000	3C00

Another solution:

```
c = 0;
while (1) {
        if (a > b) {
              a = a - b;
              c++;
        } else {
              break;
        }
}
return c;
```

0x00	SUB R7, R7, R7	0000 0111 1111 1111	07FF
0x02	SUB R0, R0, R1	0000 0110 0000 1000	0608
0x04	BREZ R1, (0x0A)	0010 1000 0100 1010	284A
0x06	INCR R7, R7	0010 0011 1100 0111	23C7
0x08	JUMP (0x02)	0011 1000 0000 0010	3802
0x0A	HALT	0011 1100 0000 0000	3C00

There can be different solutions.