Arch Lab 1

Iris Taubkin 208410969   Omri Elad 204620702

1. Warm-up: (all immediates are written in decimal base)

	OP	DST	Src0	Src1	IMM
1.1.	ADD	5	0	1	10
1.2.	SUB	5	0	1	512
1.3.	SUB	4	3	4	0
1.4.	JLT	1	3	1	<imm>

1.5. In order to load 32 constant we will divide the 32 constant into High-Bits and Low-Bits and use 1 ADD and 1 LHI:
ADD RD R0 R1 <16-LOW-BITS>
LHI   RD R0 R1 <16-HIGH-BITS>

1.6. Nested subroutines will be implemented using jumps, when return addresses will be stored in the assigned register (R7) and we will be using a Stack register for nested subroutines. Let's assign R6 as Stack register meaning they can't be used for computations. When a nested subroutine occurs, the value from the subroutine register (holding the return address) is stored in the memory, addressed by the stack register. After storing the value in Stack increases.
The stack has an assigned area in the memory. It can be assigned in the begin or the end of the address space.

Arch Lab 1

Iris Taubkin 208410969 Omri Elad 204620702

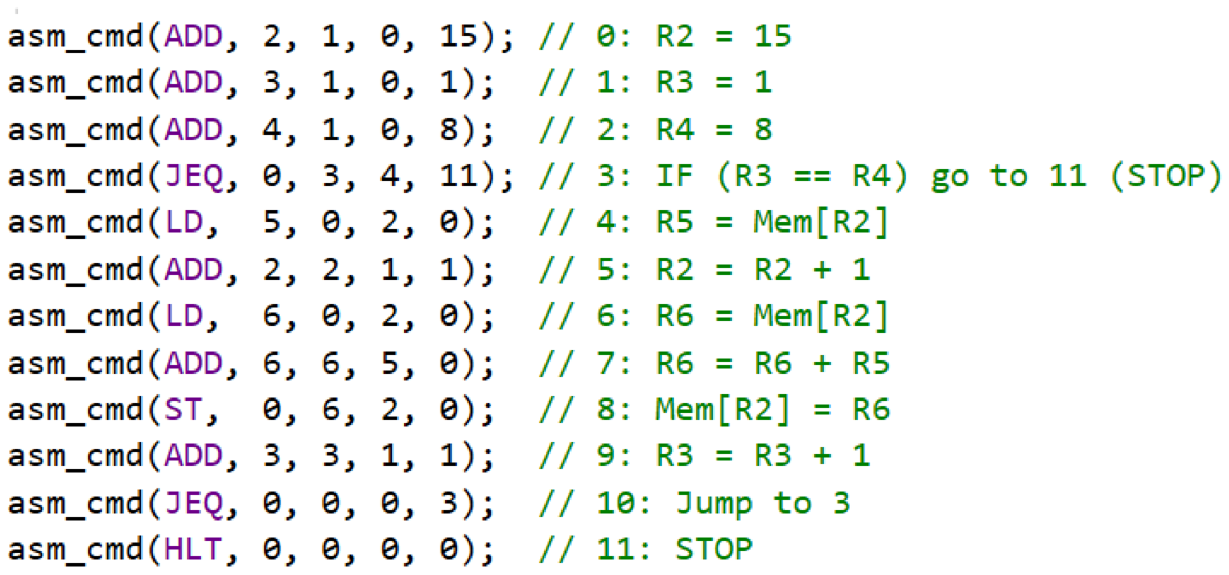
2. Example Program

    2.1. The programs run over an array of size 8, and compute the partial sum of each index.

    2.2. The inputs stored in the memory from address 15 to 22

    2.3. The outputs override the inputs, stored at the same addresses.

    2.4.



```
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) go to 11 (STOP)
asm_cmd(LD,  5, 0, 2, 0);  // 4: R5 = Mem[R2]
asm_cmd(ADD, 2, 2, 1, 1);  // 5: R2 = R2 + 1
asm_cmd(LD,  6, 0, 2, 0);  // 6: R6 = Mem[R2]
asm_cmd(ADD, 6, 6, 5, 0);  // 7: R6 = R6 + R5
asm_cmd(ST,  0, 6, 2, 0);  // 8: Mem[R2] = R6
asm_cmd(ADD, 3, 3, 1, 1);  // 9: R3 = R3 + 1
asm_cmd(JEQ, 0, 0, 0, 3);  // 10: Jump to 3
asm_cmd(HLT, 0, 0, 0, 0);  // 11: STOP
```

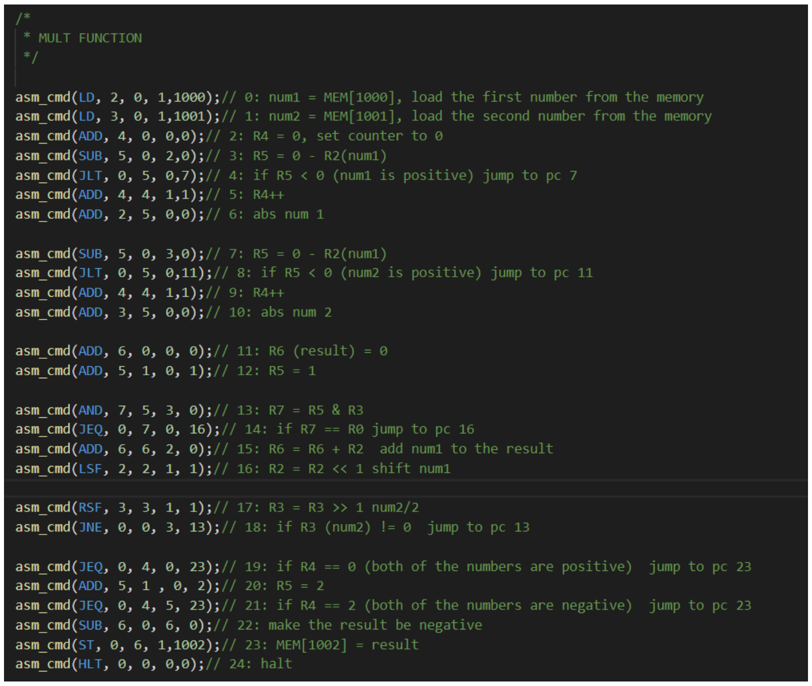
    2.5. In order to reduce memory accesses in the code, one may access in each iteration only the next items and use R6 as accumulating-addition result.
    The first memory access (for address 15) will be happening before the JEQ, hence it is possible to not call (LD, 5, 0, 2, 0) and replace call (LD, 6, 0, 2, 0) with call (LD, 5, 0, 2, 0).

Arch Lab 1

Iris Taubkin 208410969   Omri Elad 204620702

3.

```
/*
 * MULT FUNCTION
 */
asm_cmd(LD, 2, 0, 1,1000);// 0: num1 = MEM[1000], load the first number from the memory
asm_cmd(LD, 3, 0, 1,1001);// 1: num2 = MEM[1001], load the second number from the memory
asm_cmd(ADD, 4, 0, 0,0);// 2: R4 = 0, set counter to 0
asm_cmd(SUB, 5, 0, 2,0);// 3: R5 = 0 - R2(num1)
asm_cmd(JLT, 0, 5, 0,7);// 4: if R5 < 0 (num1 is positive) jump to pc 7
asm_cmd(ADD, 4, 4, 1,1);// 5: R4++
asm_cmd(ADD, 2, 5, 0,0);// 6: abs num 1
asm_cmd(SUB, 5, 0, 3,0);// 7: R5 = 0 - R2(num1)
asm_cmd(JLT, 0, 5, 0,11);// 8: if R5 < 0 (num2 is positive) jump to pc 11
asm_cmd(ADD, 4, 4, 1,1);// 9: R4++
asm_cmd(ADD, 3, 5, 0,0);// 10: abs num 2
asm_cmd(ADD, 6, 0, 0, 0);// 11: R6 (result) = 0
asm_cmd(ADD, 5, 1, 0, 1);// 12: R5 = 1
asm_cmd(AND, 7, 5, 3, 0);// 13: R7 = R5 & R3
asm_cmd(JEQ, 0, 7, 0, 16);// 14: if R7 == R0 jump to pc 16
asm_cmd(ADD, 6, 6, 2, 0);// 15: R6 = R6 + R2  add num1 to the result
asm_cmd(LSF, 2, 2, 1, 1);// 16: R2 = R2 << 1 shift num1
asm_cmd(RSF, 3, 3, 1, 1);// 17: R3 = R3 >> 1 num2/2
asm_cmd(JNE, 0, 0, 3, 13);// 18: if R3 (num2) != 0  jump to pc 13
asm_cmd(JEQ, 0, 4, 0, 23);// 19: if R4 == 0 (both of the numbers are positive)  jump to pc 23
asm_cmd(ADD, 5, 1 , 0, 2);// 20: R5 = 2
asm_cmd(JEQ, 0, 4, 5, 23);// 21: if R4 == 2 (both of the numbers are negative)  jump to pc 23
asm_cmd(SUB, 6, 0, 6, 0);// 22: make the result be negative
asm_cmd(ST, 0, 6, 1,1002);// 23: MEM[1002] = result
asm_cmd(HLT, 0, 0, 0,0);// 24: halt
```



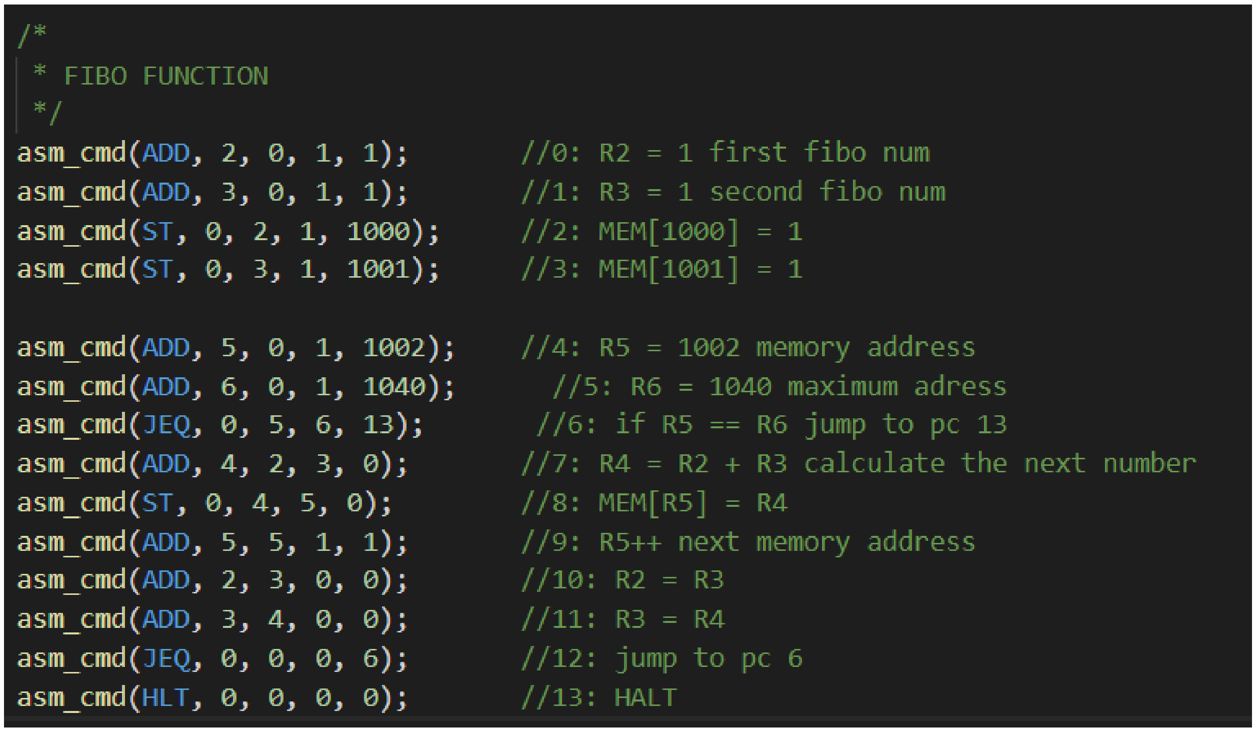
```
/*
 * MULT FUNCTION
 */

asm_cmd(LD, 2, 0, 1,1000);// 0: num1 = MEM[1000], load the first number from the memory
asm_cmd(LD, 3, 0, 1,1001);// 1: num2 = MEM[1001], load the second number from the memory
asm_cmd(ADD, 4, 0, 0,0);// 2: R4 = 0, set counter to 0
asm_cmd(SUB, 5, 0, 2,0);// 3: R5 = 0 - R2(num1)
asm_cmd(JLT, 0, 5, 0,7);// 4: if R5 < 0 (num1 is positive) jump to pc 7
asm_cmd(ADD, 4, 4, 1,1);// 5: R4++
asm_cmd(ADD, 2, 5, 0,0);// 6: abs num 1

asm_cmd(SUB, 5, 0, 3,0);// 7: R5 = 0 - R2(num1)
asm_cmd(JLT, 0, 5, 0,11);// 8: if R5 < 0 (num2 is positive) jump to pc 11
asm_cmd(ADD, 4, 4, 1,1);// 9: R4++
asm_cmd(ADD, 3, 5, 0,0);// 10: abs num 2

asm_cmd(ADD, 6, 0, 0, 0);// 11: R6 (result) = 0
asm_cmd(ADD, 5, 1, 0, 1);// 12: R5 = 1

asm_cmd(AND, 7, 5, 3, 0);// 13: R7 = R5 & R3
asm_cmd(JEQ, 0, 7, 0, 16);// 14: if R7 == R0 jump to pc 16
asm_cmd(ADD, 6, 6, 2, 0);// 15: R6 = R6 + R2  add num1 to the result
asm_cmd(LSF, 2, 2, 1, 1);// 16: R2 = R2 << 1 shift num1

asm_cmd(RSF, 3, 3, 1, 1);// 17: R3 = R3 >> 1 num2/2
asm_cmd(JNE, 0, 0, 3, 13);// 18: if R3 (num2) != 0  jump to pc 13

asm_cmd(JEQ, 0, 4, 0, 23);// 19: if R4 == 0 (both of the numbers are positive)  jump to pc 23
asm_cmd(ADD, 5, 1 , 0, 2);// 20: R5 = 2
asm_cmd(JEQ, 0, 4, 5, 23);// 21: if R4 == 2 (both of the numbers are negative)  jump to pc 23
asm_cmd(SUB, 6, 0, 6, 0);// 22: make the result be negative
asm_cmd(ST, 0, 6, 1,1002);// 23: MEM[1002] = result
asm_cmd(HLT, 0, 0, 0,0);// 24: halt
```

Arch Lab 1

Iris Taubkin 208410969   Omri Elad 204620702

4.

```
/*
 *  FIBO FUNCTION
 */
 asm_cmd(ADD, 2, 0, 1, 1);       //0: R2 = 1 first fibo num
 asm_cmd(ADD, 3, 0, 1, 1);       //1: R3 = 1 second fibo num
 asm_cmd(ST, 0, 2, 1, 1000);     //2: MEM[1000] = 1
 asm_cmd(ST, 0, 3, 1, 1001);     //3: MEM[1001] = 1
 asm_cmd(ADD, 5, 0, 1, 1002);    //4: R5 = 1002 memory address
 asm_cmd(ADD, 6, 0, 1, 1040);      //5: R6 = 1040 maximum address
 asm_cmd(JEQ, 0, 5, 6, 13);      //6: if R5 == R6 jump to pc 13
 asm_cmd(ADD, 4, 2, 3, 0);       //7: R4 = R2 + R3 calculate the next number
 asm_cmd(ST, 0, 4, 5, 0);        //8: MEM[R5] = R4
 asm_cmd(ADD, 5, 5, 1, 1);       //9: R5++ next memory address
 asm_cmd(ADD, 2, 3, 0, 0);       //10: R2 = R3
 asm_cmd(ADD, 3, 4, 0, 0);       //11: R3 = R4
 asm_cmd(JEQ, 0, 0, 0, 6);       //12: jump to pc 6
 asm_cmd(HLT, 0, 0, 0, 0);       //13: HALT
```



```
/*
 * FIBO FUNCTION
 */
asm_cmd(ADD, 2, 0, 1, 1);       //0: R2 = 1 first fibo num
asm_cmd(ADD, 3, 0, 1, 1);       //1: R3 = 1 second fibo num
asm_cmd(ST, 0, 2, 1, 1000);     //2: MEM[1000] = 1
asm_cmd(ST, 0, 3, 1, 1001);     //3: MEM[1001] = 1

asm_cmd(ADD, 5, 0, 1, 1002);    //4: R5 = 1002 memory address
asm_cmd(ADD, 6, 0, 1, 1040);      //5: R6 = 1040 maximum adress
asm_cmd(JEQ, 0, 5, 6, 13);       //6: if R5 == R6 jump to pc 13
asm_cmd(ADD, 4, 2, 3, 0);       //7: R4 = R2 + R3 calculate the next number
asm_cmd(ST, 0, 4, 5, 0);        //8: MEM[R5] = R4
asm_cmd(ADD, 5, 5, 1, 1);       //9: R5++ next memory address
asm_cmd(ADD, 2, 3, 0, 0);       //10: R2 = R3
asm_cmd(ADD, 3, 4, 0, 0);       //11: R3 = R4
asm_cmd(JEQ, 0, 0, 0, 6);       //12: jump to pc 6
asm_cmd(HLT, 0, 0, 0, 0);       //13: HALT
```