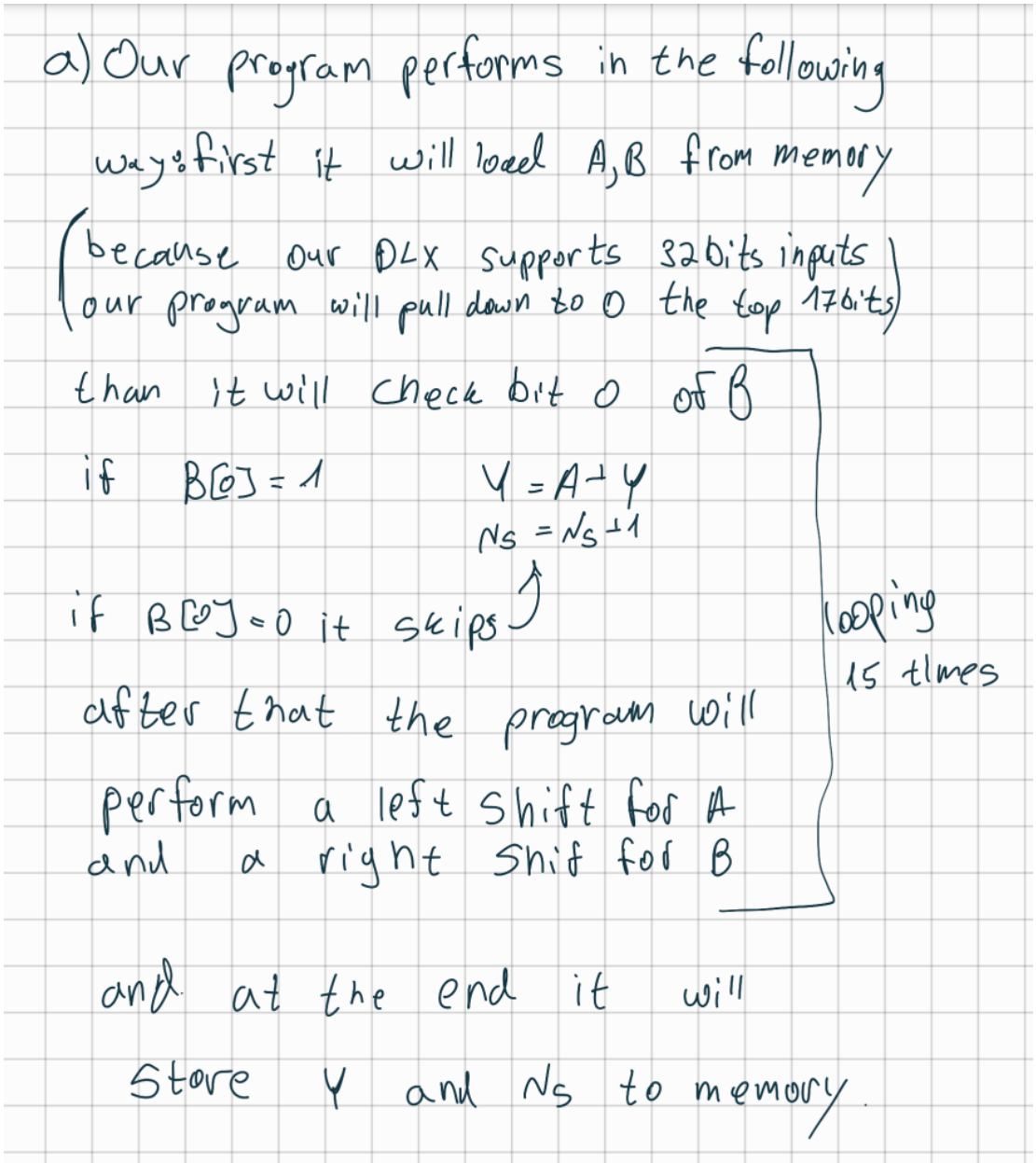
ACSL - Programming assignment

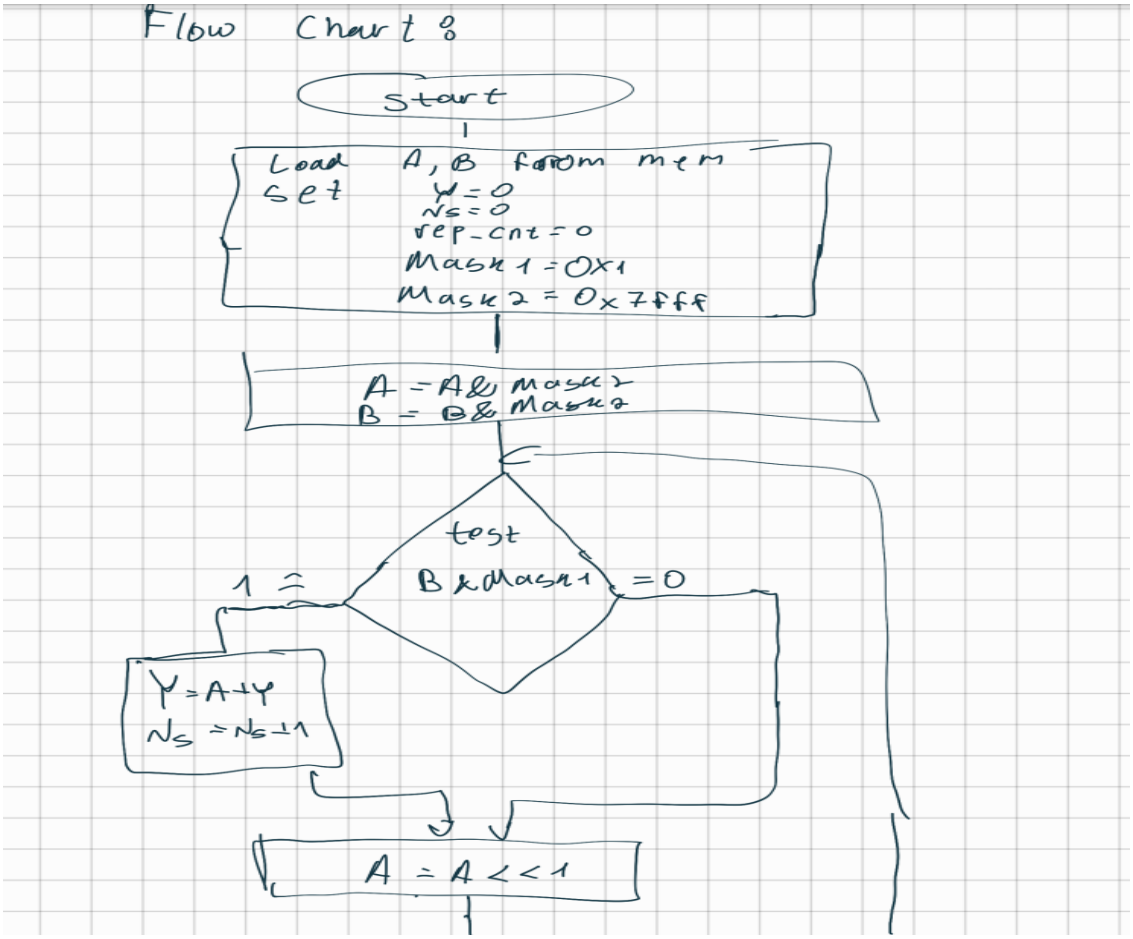
group: B2

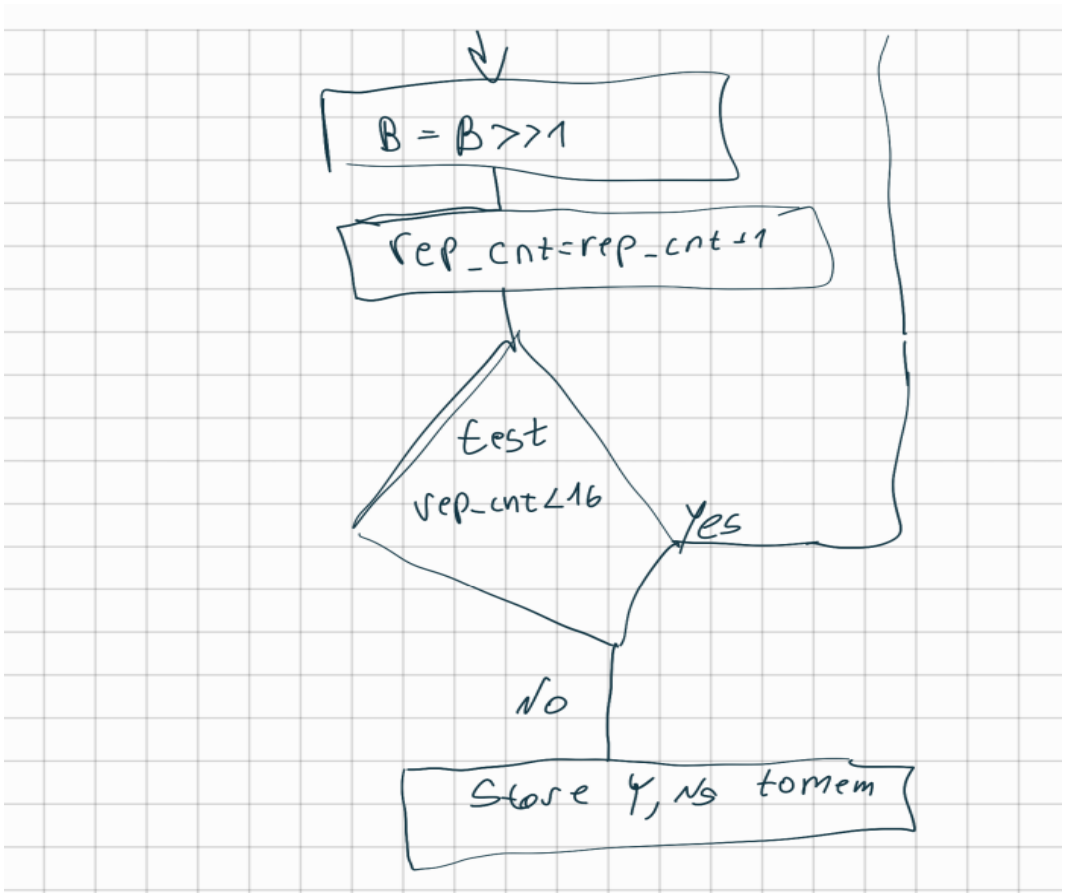
Yohai Shiloh

Yarin Koren



Flow Chart :


start
Load A, B from mem
Set Y = 0
Ns = 0
rep_cnt = 0
Mask1 = 0x1
Mask2 = 0x7fff
A = A & Mask2
B = B & Mask2
test
B & Mask1 = 0
1 ≠
Y = A + Y
Ns = Ns + 1
A = A << 1


B = B >> 1
rep_cnt = rep_cnt + 1
test
rep_cnt < 16
Yes
No
Store Y, Ns to mem

Memory organization:

Since, A at address 0x3
B at address 0x4
Y at address 0x5
N's at address 0x6

in our program addresses 0-1 contains some instruction, address 2 contains branch instruction to address 7

addresses 3,4 contains A, B

at addresses 5, 6 we save 1 word each for Y and N's and in the following addresses we have the rest of the code

The Code:

```
pc =0x0 *pc starts from 0

start:          lw R1 R0 A              *R1=mem[A]
                lw R2 R0 B              *R2=mem[B]
                beqz R0 continue        *jump to pc 0x7
A:              dc 0x22                 * mem[A] = 0x22
B:              dc 0x1b                 *mem[B] = 0x1b
Y:              ds 1                    *save 1 word in memory in address 0x5 for Y
Ns:             ds 1                    *save 1 word in memory in address 0x6 for Ns
continue:       addi R8 R0 0x7fff       *R8 = 0x7fff used to make sure we multiply only the first 15 bits
                and R1 R1 R8            *R1 = R1&0x7fff zeroing the top 17 bits
                and R2 R2 R8            *R2 = R2&0x7fff zeroing the top 17 bits
                addi R3 R0 0            *R3 = 0  => Y = 0
                addi R4 R0 0            *R4 = 0 => rep_cnt = 0
                addi R5 R0 1            *R5 = 1 used to take bit0 from B
                addi R6 R0 0            *R6 = 0 => Ns = 0
loop:           and R7 R5 R2            *R7 = B[0]
                beqz R7 shift           *if B[0] = 0 jump to shift
                add R3 R3 R1            *Y = Y + A
                addi R6 R6 1            *Ns = Ns + 1
shift:          slli R1 R1              *A = A << 1
                srli R2 R2              *B = B >>1
                addi R4 R4 1            *rep_cnt++
                slti R7 R4 0x10         *checks if num of rep less than 16
                bnez R7 loop            *if rep < 16 jump to loop
                sw R3 R0 Y              *store Y in mem 0x5
                sw R6 R0 Ns             *store Ns in mem 0x6
                halt                    *program ends
```

.lst file:

```
0x00000000: 0x8C010003  start:                          lw R1 R0 A
0x00000001: 0x8C020004                                  lw R2 R0 B
0x00000002: 0x101F0004                                  beqz R0 continue
0x00000003: 0x87650006  A:                              dc 0x87650006
0x00000004: 0x88880009  B:                              dc 0x88880009
0x00000005: 0x00000000  Y:              dc 0x0
0x00000006: 0x00000000  Ns:             dc 0x0
0x00000007: 0x2C087FFF  continue:                       addi R8 R0 0x7fff
0x00000008: 0x00280826                                  and R1 R1 R8
0x00000009: 0x00481026                                  and R2 R2 R8
0x0000000A: 0x2C030000                                  addi R3 R0 0
0x0000000B: 0x2C040000                                  addi R4 R0 0
0x0000000C: 0x2C050001                                  addi R5 R0 1
0x0000000D: 0x2C060000                                  addi R6 R0 0
0x0000000E: 0x00A23826  loop:                           and R7 R5 R2
0x0000000F: 0x10FF0002                                  beqz R7 shift
0x00000010: 0x00611823                                  add R3 R3 R1
0x00000011: 0x2CC60001                                  addi R6 R6 1
0x00000012: 0x003F0800  shift:                          slli R1 R1
0x00000013: 0x005F1002                                  srli R2 R2
0x00000014: 0x2C840001                                  addi R4 R4 1
0x00000015: 0x70870010                                  slti R7 R4 0x10
0x00000016: 0x14FFFFF7                                  bnez R7 loop
0x00000017: 0xAC030005                                  sw R3 R0 Y
0x00000018: 0xAC060006                                  sw R6 R0 Ns
0x00000019: 0xFFFF203A                                  halt


Label Report:
start: 3 D:
a: 6 D: 3
b: 7 D: 4
y: 8 D: 26
ns: 9 D: 27
continue: 10 D: 5
loop: 17 D: 25
shift: 21 D: 18
*XML file date: Wed 20/6/2012 6:49:12
```

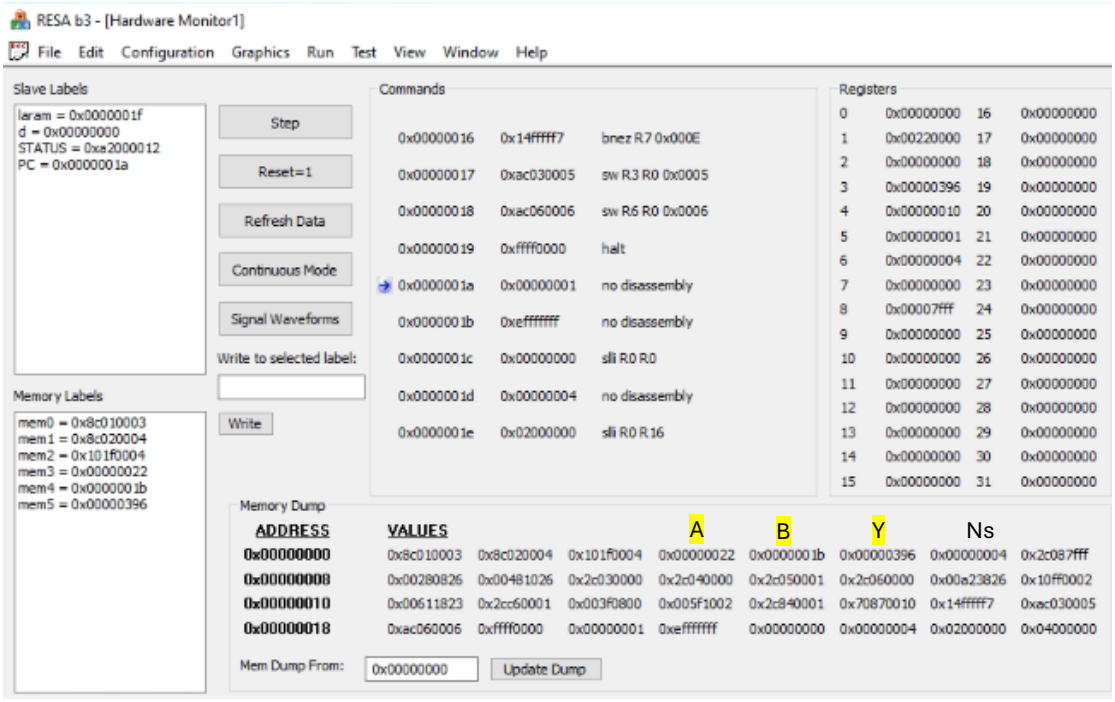
.cod file:

```
.CODE
0x00000000
0x00000003
0x8C010003
0x8C020004
0x101F0004
.DATA
0x00000003
0x00000004
0x87650006
0x88880009
0x00000000
0x00000000
.CODE
0x00000007
0x00000013
0x2C087FFF
0x00280826
0x00481026
0x2C030000
0x2C040000
0x2C050001
0x2C060000
0x00A23826
0x10FF0002
0x00611823
0x2CC60001
0x003F0800
0x005F1002
0x2C840001
0x70870010
0x14FFFFF7
0xAC030005
0xAC060006
0xFFFF203A
.DS
0X52 XML file date: XML file date: Wed 20/6/2012 6:49:12
```

Result with the inputs from the example:

A = 0x22

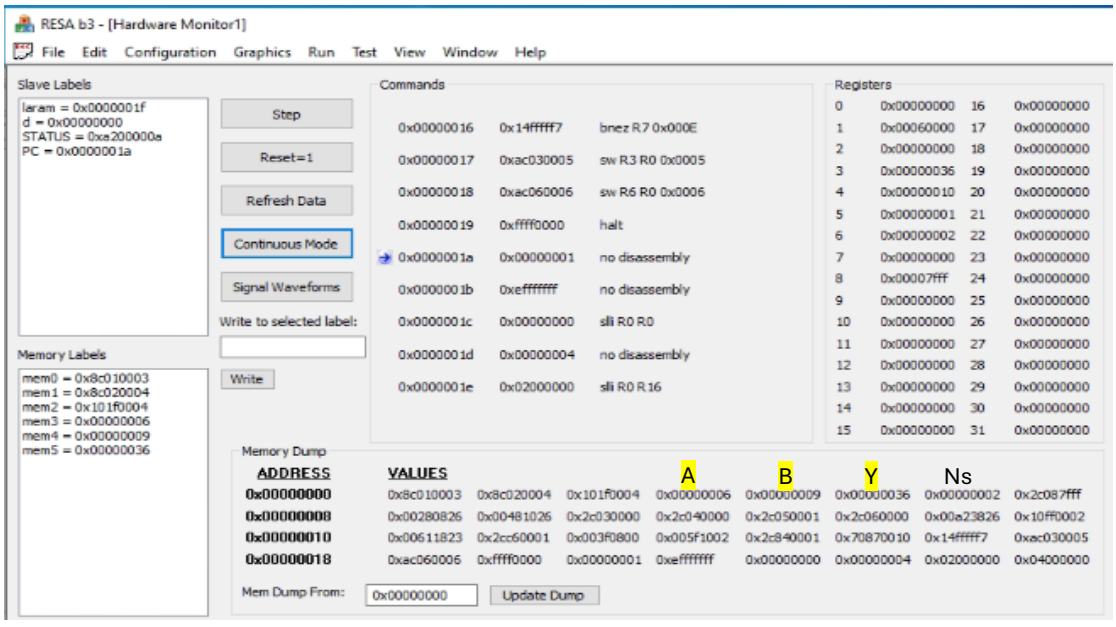
B = 0x1b



Result with our inputs:

A = 0x6

B =0x9



we see that all the inputs and outputs are in their requested addresses and the values we expect to receive are correct.