

ACSL - Programming assignment

group: B2

Yohai Shiloh

Yarin Koren

a) Our program performs in the following way: first it will load A, B from memory

(because our DLX supports 32 bits inputs)
(our program will pull down to 0 the top 17 bits)

then it will check bit 0 of B

if $B[0] = 1$

$$Y = A + Y$$

$$N_s = N_s + 1$$

if $B[0] = 0$ it skips

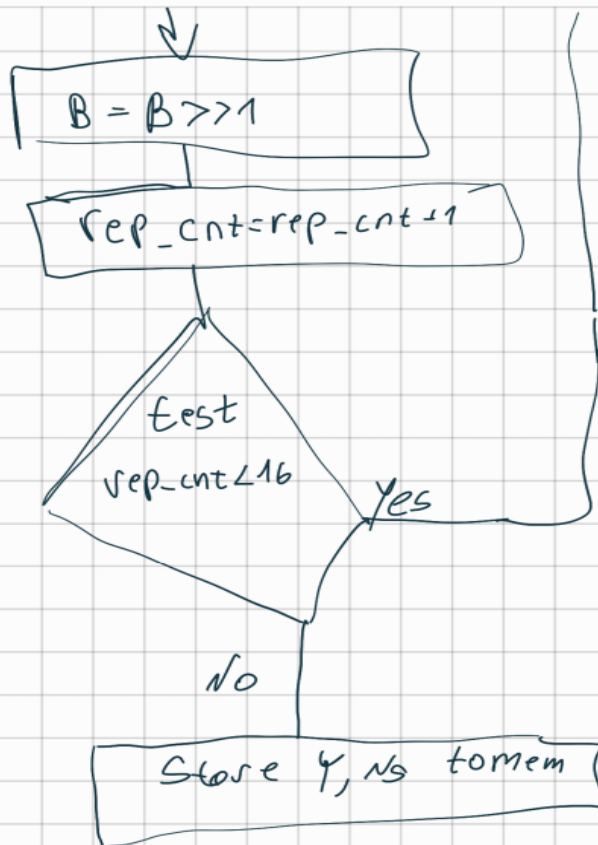
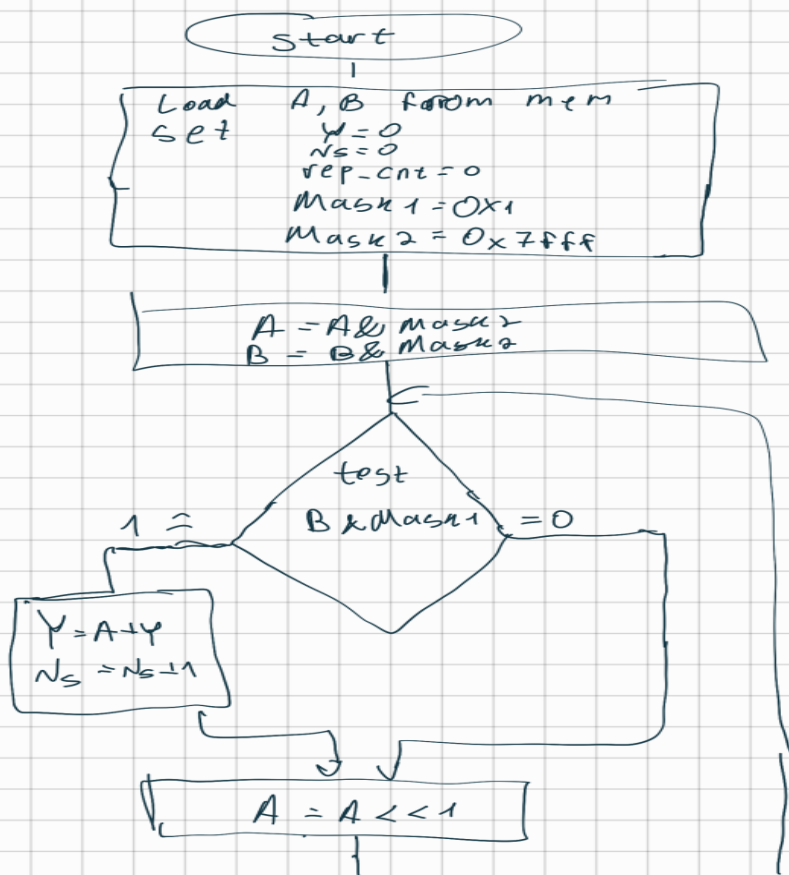
after that the program will
perform a left shift for A
and a right shift for B

looping
15 times

and at the end it will

store Y and N_s to memory.

Flow Chart 2



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:	0x14FFFFFF		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
0x14FFFFFF7
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

The screenshot shows the RESA b3 - [Hardware Monitor1] interface. The 'Slave Labels' section on the left contains:
lram = 0x0000001f
d = 0x00000000
STATUS = 0xa2000012
PC = 0x0000001a
The 'Memory Labels' section contains:
mem0 = 0x8c010003
mem1 = 0x8c020004
mem2 = 0x101f0004
mem3 = 0x00000022
mem4 = 0x0000001b
mem5 = 0x00000396
The 'Commands' section shows a list of assembly instructions, with the current instruction being 'no disassembly' at address 0x0000001a. The 'Registers' section shows a list of registers, with the current register being 0x00000000. The 'Memory Dump' section shows a table of memory addresses and values, with the current address being 0x00000000. The 'Mem Dump From' field is set to 0x00000000.

ADDRESS	VALUES	A	B	Y	Ns
0x00000000	0x8c010003 0x8c020004 0x101f0004 0x00000022 0x0000001b 0x00000396 0x00000004 0x2c087fff				
0x00000008	0x00280826 0x00481026 0x2c030000 0x2c040000 0x2c050001 0x2c060000 0x00a23826 0x10ff0002				
0x00000010	0x00611823 0x2cc60001 0x003f0800 0x005f1002 0x2c840001 0x70870010 0x14fffff7 0xac030005				
0x00000018	0xac060006 0xffff0000 0x00000001 0xffffffff 0x00000000 0x00000004 0x02000000 0x40000000				

Result with our inputs:

A = 0x6

B = 0x9

The screenshot shows the RESA b3 - [Hardware Monitor1] interface with our inputs. The 'Slave Labels' section contains:
lram = 0x0000001f
d = 0x00000000
STATUS = 0xa200000a
PC = 0x0000001a
The 'Memory Labels' section contains:
mem0 = 0x8c010003
mem1 = 0x8c020004
mem2 = 0x101f0004
mem3 = 0x00000006
mem4 = 0x00000009
mem5 = 0x00000036
The 'Commands' section shows a list of assembly instructions, with the current instruction being 'no disassembly' at address 0x0000001a. The 'Registers' section shows a list of registers, with the current register being 0x00000000. The 'Memory Dump' section shows a table of memory addresses and values, with the current address being 0x00000000. The 'Mem Dump From' field is set to 0x00000000.

ADDRESS	VALUES	A	B	Y	Ns
0x00000000	0x8c010003 0x8c020004 0x101f0004 0x00000006 0x00000009 0x00000036 0x00000002 0x2c087fff				
0x00000008	0x00280826 0x00481026 0x2c030000 0x2c040000 0x2c050001 0x2c060000 0x00a23826 0x10ff0002				
0x00000010	0x00611823 0x2cc60001 0x003f0800 0x005f1002 0x2c840001 0x70870010 0x14fffff7 0xac030005				
0x00000018	0xac060006 0xffff0000 0x00000001 0xffffffff 0x00000000 0x00000004 0x02000000 0x40000000				

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