Computation structures: tutorial

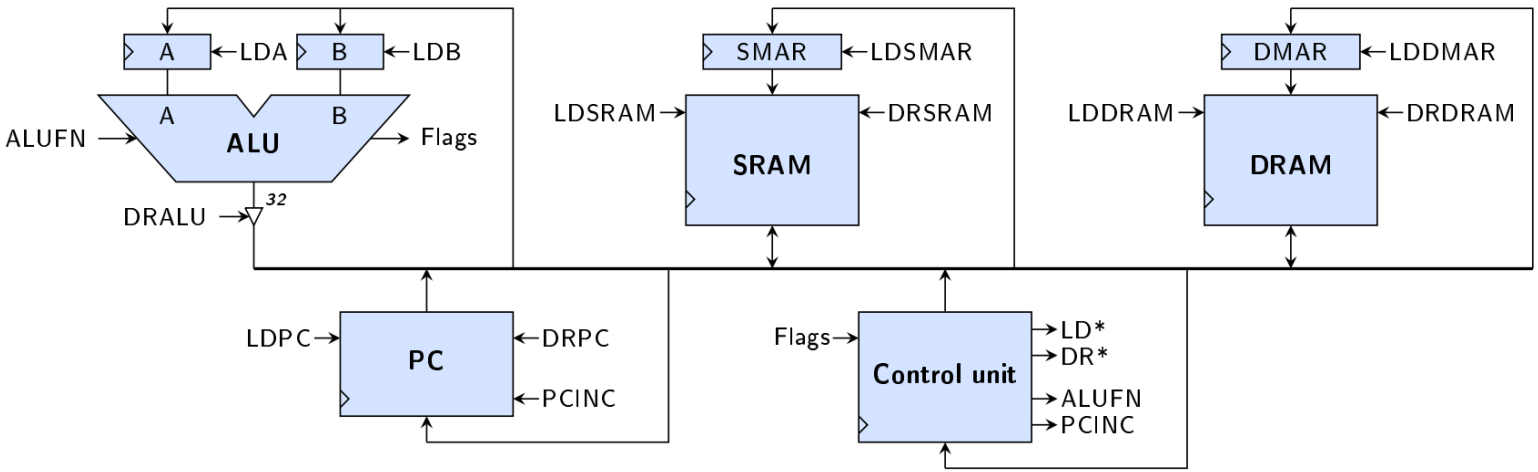
Microcode (μ-code) (part 1)

Romain Mormont

Montefiore Institute, University of Liège, Belgium

Q1 – 2020-2021

β-machine with bus



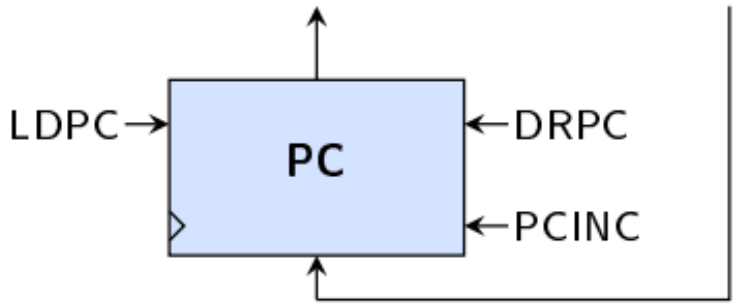
- new architecture, a Von Neumann machine
- components exchange information through a unique and shared bus
- orchestrated by the control logic using microcode (μ-code)
- one instruction takes several clock cycles to execute

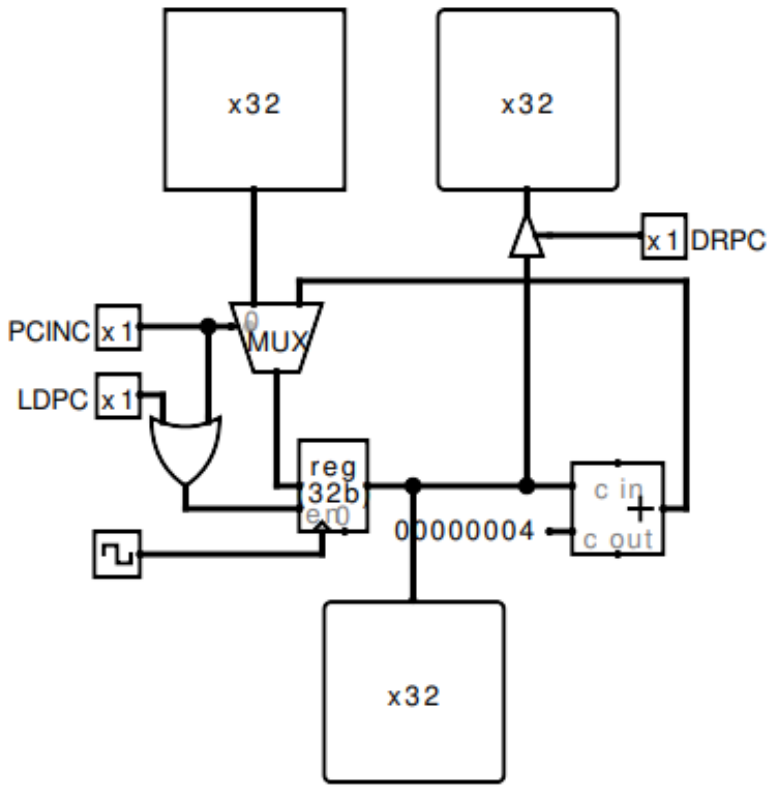
Program counter (PC)

Basically a register that stores the address of the **next** instruction to be executed.

Input signals:

- `LDPC`: update PC with value from the bus
- `DRPC`: put PC value on the bus
- `PCINC` (`PC+`): increment the value of PC (+4)





Arithmetic logic unit (ALU)

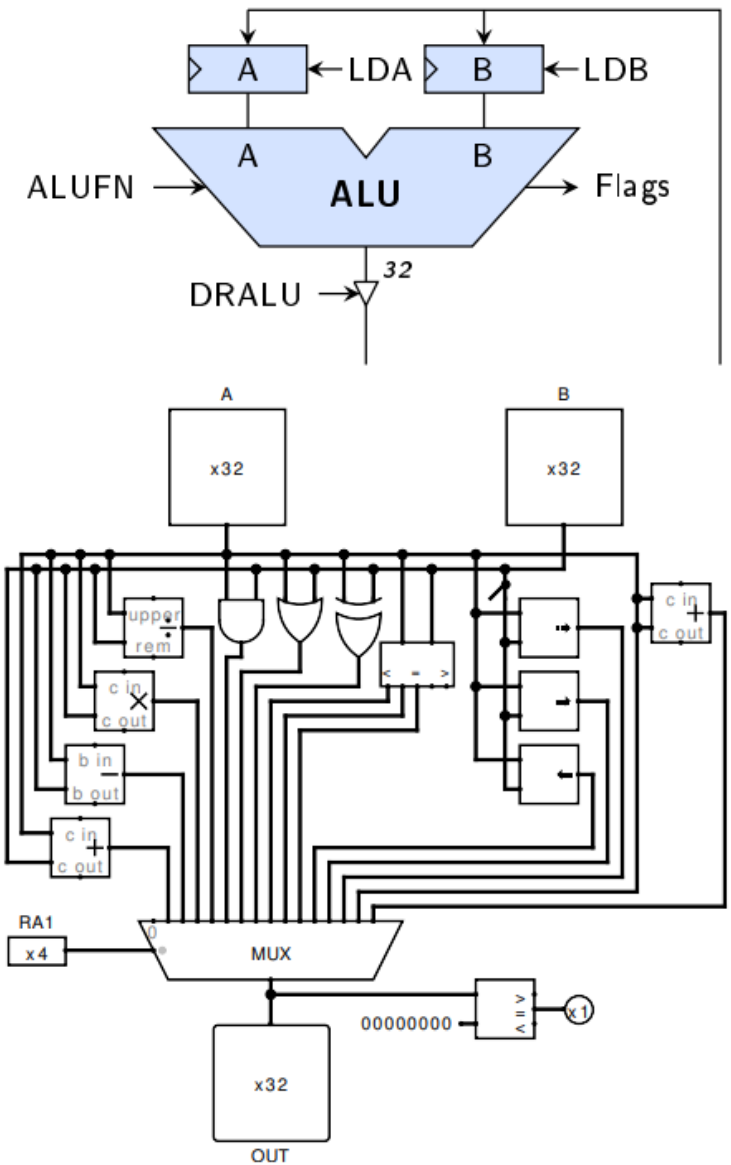
Similar as before, with few updates.

Input signals:

- `LDA`: load value from the bus in register A
- `LDB`: load value from the bus in register B
- `DRALU`: put ALU output on the bus
- `ALUFN`: ALU function code

Flags:

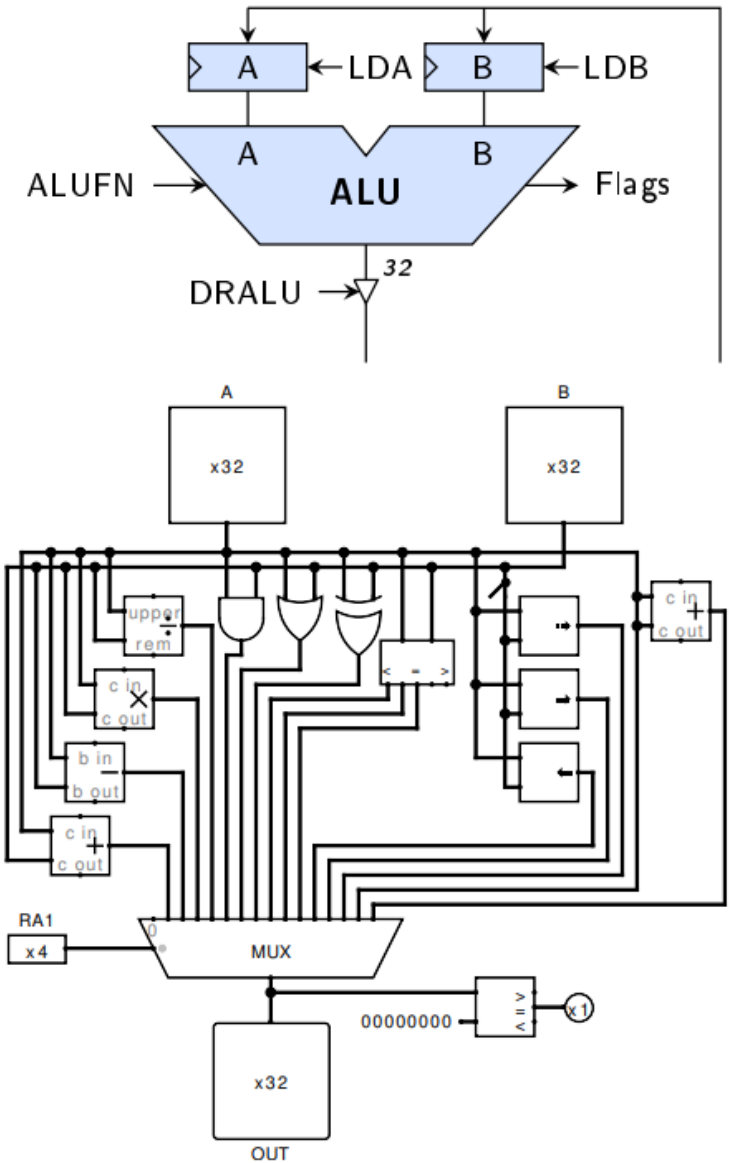
- `E`: set to 1 when the ALU output is equal to 0. To be used by the control logic for conditional execution.



Arithmetic logic unit (ALU)

Function		ALUFN
$*$	$*$	0b0000
$A + B$	addition	0b0001
$A - B$	substr.	0b0010
$A \times B$	multipl.	0b0011
$A \div B$	division	0b0100
$A \,\&\, B$	bitwise and	0b0101
$A \mid B$	bitwise or	0b0110
$A \oplus B$	bitwise xor	0b0111
$A < B$	cmplt	0b1000
$A \leqslant B$	cmple	0b1001
$A == B$	cmpeq	0b1010
$A \ll B$	shift left	0b1011
$A \gg B$	shift right	0b1100
$A \gg^{31} B$	sra	0b1101
A	give A	0b1110
$A + A$	double A	0b1111

Available functions

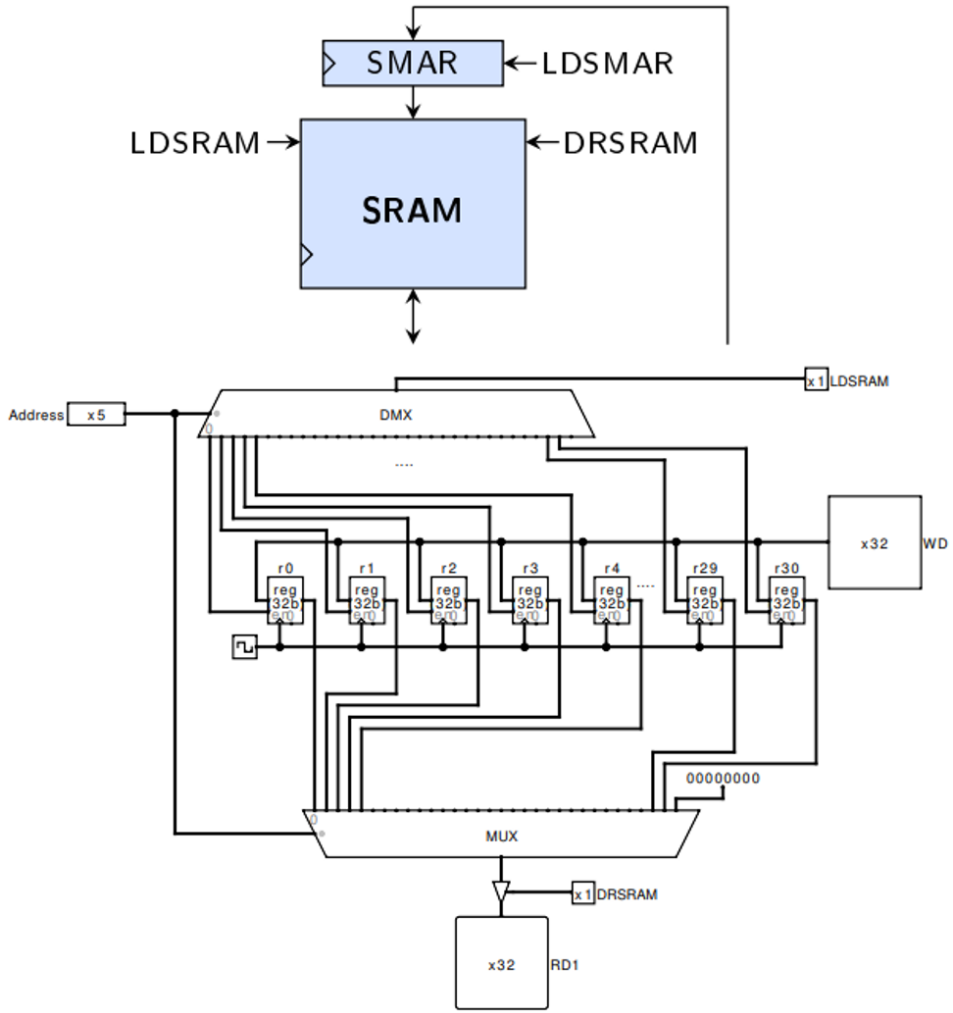


Static RAM (SRAM)

Simplified version of the register file. Only **one register address** and input/output **at a time**. Address is stored in the Static Memory Address Register (SMAR).

Input signals:

- LDSMAR: load the address of the register from the bus
- LDSRAM: set the value of the current register (defined by SMAR) from the bus
- DRSRAM: put the value of the current register (defined by SMAR) on the bus

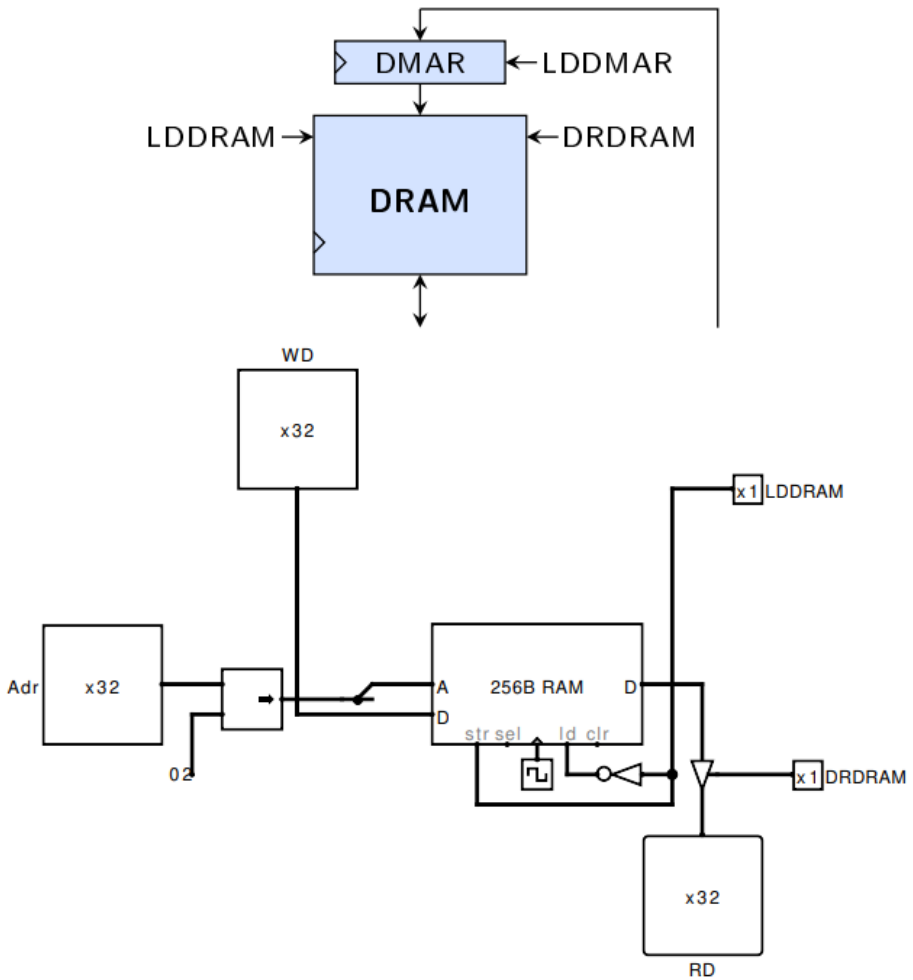


Dynamic RAM (DRAM)

Stores both code and data. Similar interactions as for SRAM.

Input signals:

- LDDMAR: load the memory address from the bus
- LDDRAM: set the value at the current memory address (defined by DMAR) from the bus
- DRDRAM: put the value at the current memory address (defined by DMAR) on the bus

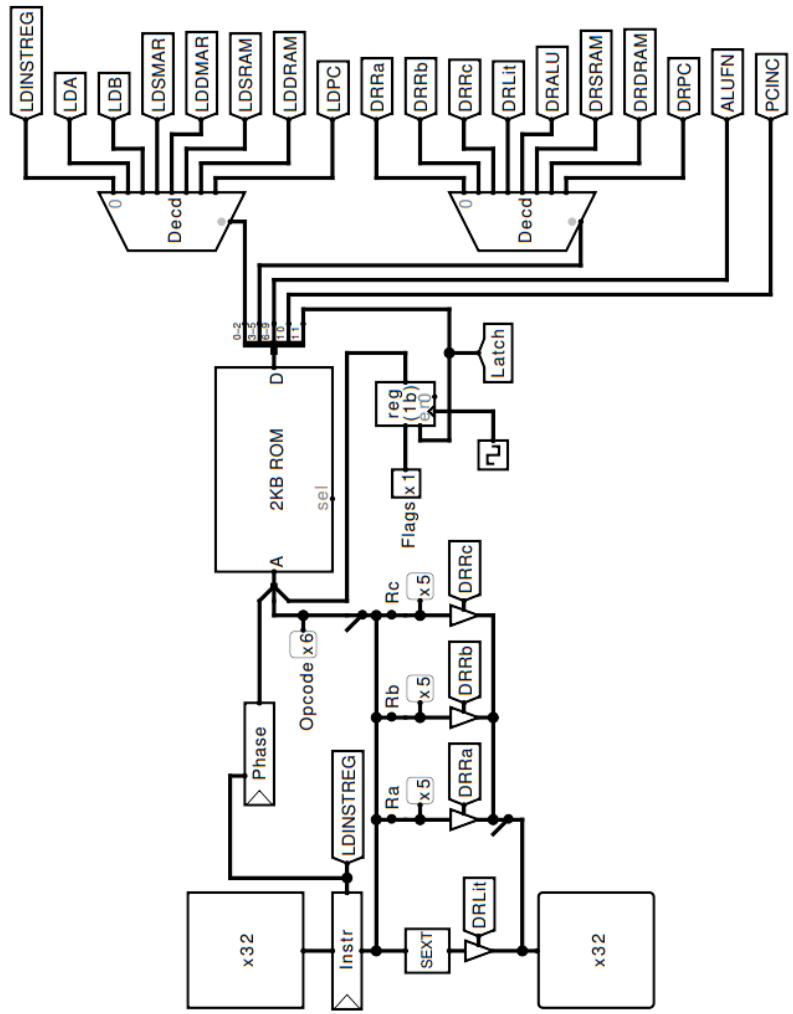


Control unit

Brain of the machine. Controls how the components interacts with each other to execute instructions.

The current instruction is stored in `INSTREG`. An instruction is executed in several clock cycles. Each step/clock cycle of an instruction execution is called a **phase** and the phase counter tracks them.

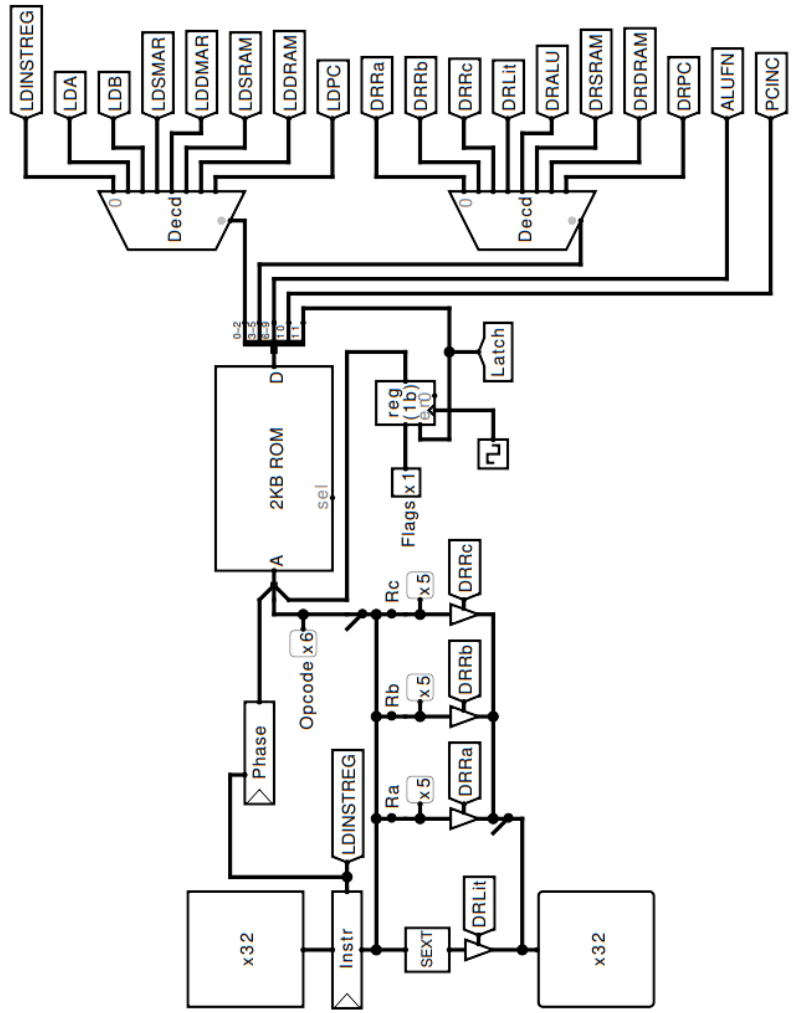
The last phases of each instruction consists in fetching and loading the next instruction in `INSTREG` which resets the phase counter.



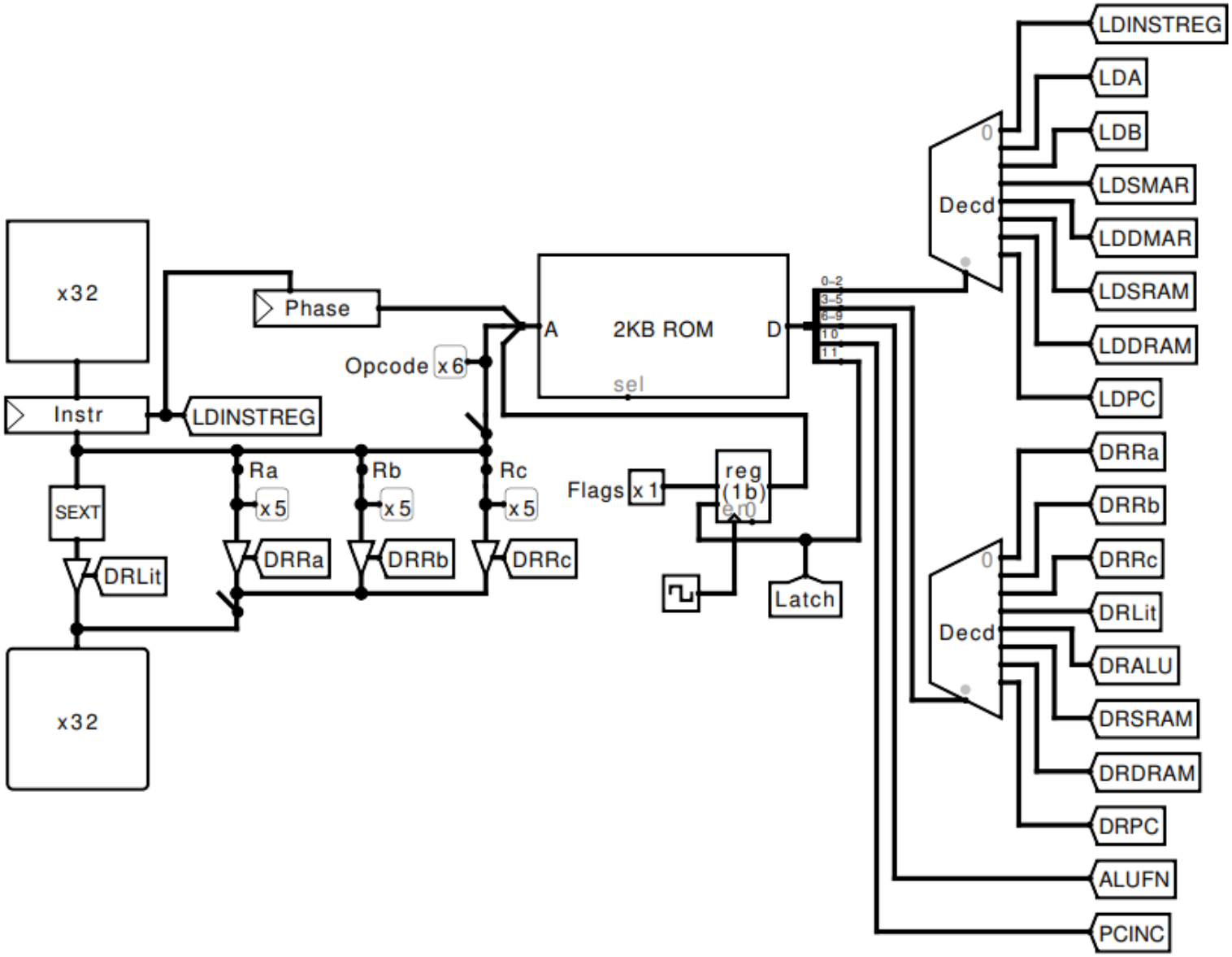
Control unit

Behavior of the machine is encoded as μ-code stored in the control ROM. An address of the ROM encodes the state of the machine (current opcode, phase, flags) and maps it to actions (*i.e.* control signals: load, drive, ALU function...) to be performed by the machine.

Also features circuits for extracting from instructions values of `Ra`, `Rb`, `Rc` and `Lit`.



Control unit



Control unit (signals)

Output signals:

- Control logic:
  - `DRLit`: put the (sign-extended) literal on the bus
  - `DRRa`: put the address of register `Ra` on the bus
  - `DRRb`: put the address of register `Rb` on the bus
  - `DRRc`: put the address of register `Rc` on the bus
  - `LDINSTREG`: set the new instruction from the bus
  - `Latch` (`LF`): latch ALU flag (save its current value into the flag register)
- ALU: `DRALU`, `LDA`, `LDB`, `ALUFN`
- static RAM: `DRSRAM`, `LDSRAM`, `LDSMAR`
- dynamic RAM: `DRDRAM`, `LDDRAM`, `LDDMAR`
- Program counter: `PC+`/`PCINC`, `LDPC`, `DRPC`

Exercice 1

Instruction BEQ.

E	Opcode	Phase	LD	DR	ALUFN	PC+	Latch	Value
0/1	011101	0000	LDSMAR/011	DRRa/000	- (0000)	0	0	000000000011
0/1	011101	0001	LDA/001	DRSRAM/101	- (0000)	0	0	000000101001
0/1	011101	0010	LDA/001	DRALU/100	A (1110)	0	1	101110100001
0/1	011101	0011	LDSMAR/011	DRRc/010	- (0000)	0	0	000000010011
0/1	011101	0100	LDSRAM/101	DRPC/111	- (0000)	0	0	000000111101
0	011101	0101	LDDMAR/100	DRPC/111	- (0000)	1	0	010000111100
0	011101	0110	LDINSTREG/000	DRDRAM/110	- (0000)	0	0	000000110000
1	011101	0101	LDA/001	DRlit/011	- (0000)	0	0	000000011001
1	011101	0110	LDA/001	DRALU/100	2*A (1111)	0	0	001111100001
1	011101	0111	LDA/001	DRALU/100	2*A (1111)	0	0	001111100001
1	011101	1000	LDB/010	DRPC/111	- (0000)	0	0	000000111010
1	011101	1001	LDPC/111	DRALU/100	ADD (0001)	0	0	000001100111
1	011101	1010	LDDMAR/100	DRPC/111	- (0000)	1	0	010000111100
1	011101	1011	LDINSTREG/000	DRDRAM/110	- (0000)	0	0	000000110000

Exercice 2

Instruction `SWAPIFZ`, swap if zero.

```
SWAPIFZ(Ra,Rb,Rc):
  PC <- PC + 4
  COND <- Reg[Rc]
  if COND == 0 then
    TMP <- Reg[Ra]
    Reg[Ra] <- Reg[Rb]
    Reg[Rb] <- TMP
```

Exercice 2

Instruction SWAPIFZ with basic ALU functions. Possibly not working according to the implementation of the overflow handling.

E	Opcode	Phase	LD	DR	ALUFN	PC+	Latch	Value
0/1	100111	0000	LDSMAR/011	DRRc/010	- (0000)	0	0	000000010011
0/1	100111	0001	LDA/001	DRSRAM/101	- (0000)	0	0	000000101001
0/1	100111	0010	LDA/001	DRALU/100	A (1110)	0	1	101110100001
0	100111	0011	LDDMAR/100	DRPC/111	- (0000)	1	0	010000111100
0	100111	0100	LDINSTREG/000	DRDRAM/110	- (0000)	0	0	000000110000
1	100111	0011	LDSMAR/011	DRRa/000	- (0000)	0	0	000000000011
1	100111	0100	LDA/001	DRSRAM/101	- (0000)	0	0	000000101001
1	100111	0101	LDSMAR/011	DRRb/001	- (0000)	0	0	000000001011
1	100111	0110	LDB/010	DRSRAM/101	- (0000)	0	0	000000101010
1	100111	0111	LDSRAM/101	DRALU/100	A (1110)	0	0	001110100101
1	100111	1000	LDSMAR/011	DRRa/000	- (0000)	0	0	000000000011
1	100111	1001	LDA/001	DRALU/100	ADD (0001)	0	0	000001100001
1	100111	1010	LDB/010	DRSRAM/101	- (0000)	0	0	000000101010
1	100111	1011	LDSRAM/101	DRALU/100	SUB (0010)	0	0	000010100101
1	100111	1100	LDDMAR/100	DRPC/111	- (0000)	1	0	010000111100
1	100111	1101	LDINSTREG/000	DRDRAM/110	- (0000)	0	0	000000110000

Exercice 2 (bis)

Instruction SWAPIFZ using a new ALU function B (0b0000) for which the ALU outputs the value stored in register B.

E	Opcode	Phase	LD	DR	ALUFN	PC+	Latch	Value
0/1	100111	0000	LDSMAR/011	DRRc/010	- (0000)	0	0	000000010011
0/1	100111	0001	LDA/001	DRSRAM/101	- (0000)	0	0	000000101001
0/1	100111	0010	LDA/001	DRALU/100	A (1110)	0	1	101110100001
0	100111	0011	LDDMAR/100	DRPC/111	- (0000)	1	0	010000111100
0	100111	0100	LDINSTREG/000	DRDRAM/110	- (0000)	0	0	000000110000
1	100111	0011	LDSMAR/011	DRRa/000	- (0000)	0	0	000000000011
1	100111	0100	LDA/001	DRSRAM/101	- (0000)	0	0	000000101001
1	100111	0101	LDSMAR/011	DRRb/001	- (0000)	0	0	000000001011
1	100111	0110	LDB/010	DRSRAM/101	- (0000)	0	0	000000101010
1	100111	0111	LDSRAM/101	DRALU/100	A (1110)	0	0	001110100101
1	100111	1000	LDSMAR/011	DRRa/000	- (0000)	0	0	000000000011
1	100111	1001	LDSRAM/101	DRALU/100	B (0000)	0	0	000000100101
1	100111	1010	LDDMAR/100	DRPC/111	- (0000)	1	0	010000111100
1	100111	1011	LDINSTREG/000	DRDRAM/110	- (0000)	0	0	000000110000

Exercice 3

Instruction `JMPI`, Jump Indirect.

```
JMPI(Ra,Lit,Rc):
  PC <- PC + 4
  Reg[Rc] <- PC
  PC <- Mem[Reg[Ra] + Lit]
```

Exercice 3

Instruction JMPI.

E	Opcode	Phase	LD	DR	ALUFN	PC+	Latch	Value
0/1	101011	0000	LDB/010	DRlit/011	- (0000)	0	0	000000011010
0/1	101011	0001	LDSMAR/011	DRRa/000	- (0000)	0	0	000000000011
0/1	101011	0010	LDA/001	DRSRAM/101	- (0000)	0	0	000000101001
0/1	101011	0011	LDSMAR/011	DRRc/010	- (0000)	0	0	000000010011
0/1	101011	0100	LDSRAM/101	DRPC/111	- (0000)	0	0	000000111101
0/1	101011	0101	LDDMAR/100	DRALU/100	ADD (0001)	0	0	000001100100
0/1	101011	0110	LDPC/111	DRDRAM/110	- (0000)	0	0	000000110111
0/1	101011	0111	LDDMAR/100	DRPC/111	- (0000)	1	0	010000111100
0/1	101011	1000	LDINSTREG/000	DRDRAM/110	- (0000)	0	0	000000110000

Exercice 4

Instruction BUZ, Branch Under Zero.

```
BUZ(Ra,Lit,Rc):
  PC <- PC + 4
  EA <- PC + 4 * SEXT(Lit)
  TMP <- Reg[Ra]
  Reg[Rc] <- PC
  if TMP < 0 then PC <- EA
```

Exercice 4

Instruction BUZ.

E	Opcode	Phase	LD	DR	ALUFN	PC+	Latch	Value
0/1	101111	0000	LDSMAR/011	DRRa/000	- (0000)	0	0	000000000011
0/1	101111	0001	LDA/001	DRSRAM/101	- (0000)	0	0	000000101001
0/1	101111	0010	LDB/010	DRSRAM/101	- (0000)	0	0	000000101010
0/1	101111	0011	LDB/010	DRALU/100	SUB (0010)	0	0	000010100010
0/1	101111	0100	LDSMAR/011	DRRc/010	- (0000)	0	0	000000010011
0/1	101111	0101	LDSRAM/101	DRPC/111	- (0000)	0	0	000000111101
0/1	101111	0110	LDB/010	DRALU/100	CMPLT (1101)	0	1	101101100010
0	101111	0111	LDA/001	DRlit/011	- (0000)	0	0	000000011001
0	101111	1000	LDA/001	DRALU/100	2*A (1111)	0	0	001111100001
0	101111	1001	LDA/001	DRALU/100	2*A (1111)	0	0	001111100001
0	101111	1010	LDB/010	DRPC/111	- (0000)	0	0	000000111010
0	101111	1011	LDPC/111	DRALU/100	ADD (0001)	0	0	000001100111
0	101111	1100	LDDMAR/100	DRPC/111	- (0000)	1	0	010000111100
0	101111	1101	LDINSTREG/000	DRDRAM/110	- (0000)	0	0	000000110000
1	101111	0111	LDDMAR/100	DRPC/111	- (0000)	1	0	010000111100
1	101111	1000	LDINSTREG/000	DRDRAM/110	- (0000)	0	0	000000110000