

# Computation structures

# Introduction

19 Sept. 2017

Romain Mormont

Office I.128

*Email:* [r.mormont@ulg.ac.be](mailto:r.mormont@ulg.ac.be)

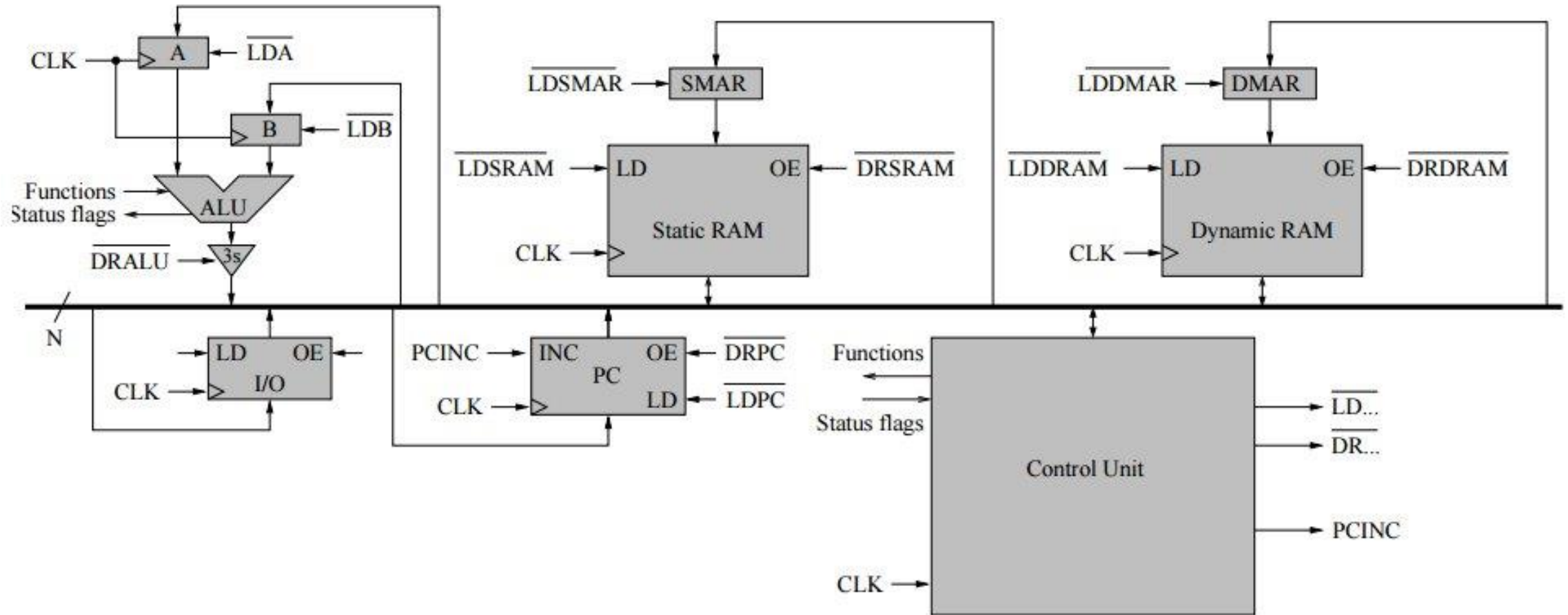
# General information

- Course webpage:  
<http://www.montefiore.ulg.ac.be/~rmormont/?rpath=/info0012>
- Tutorial every Tuesday, here (R3) and now (~ from 16h to 18h)
- Grading:
  - Two projects:
    1.  $\beta$ -assembly: due for first week of November (to be confirmed)
    2. Parallel programming
  - Written exam in January

# Computation structures

## Tutorial 1: $\mu$ -code for ULg01

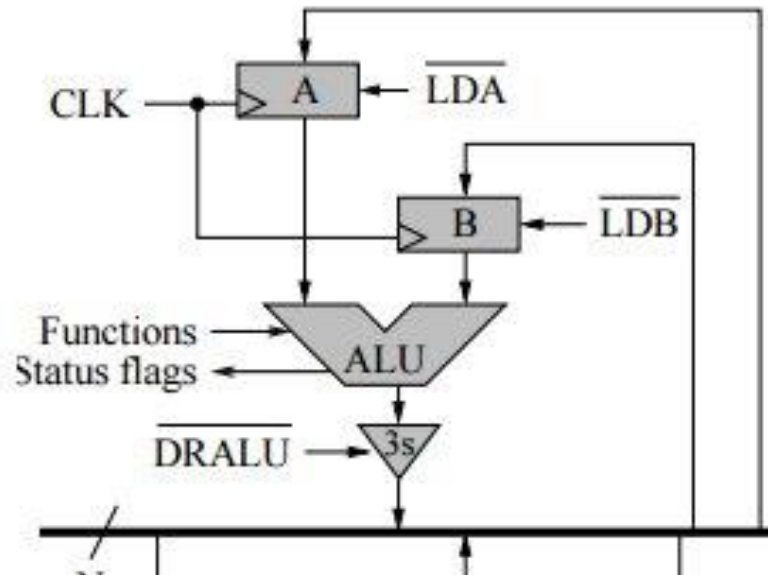
# ULg01: an implementation of the $\beta$ -machine



# ULg01: data path

The data path includes:

- **ALU circuit: Arithmetic Logic Unit**
- **Static memory (DRAM):** similar to your computer's processor registers
- **Dynamic memory (SRAM):** similar to your computer's RAM



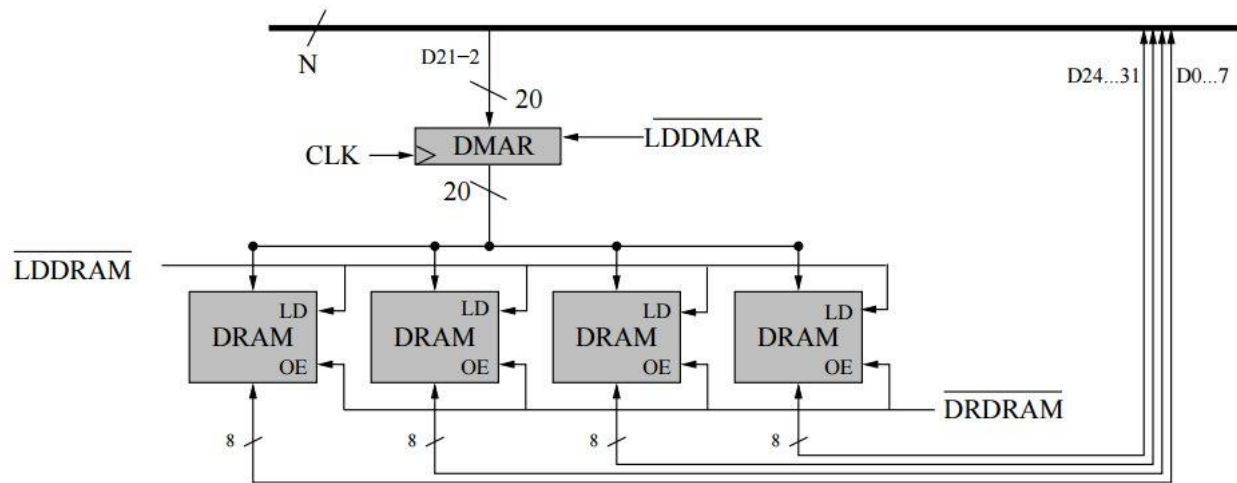
## ALU:

- Two registers **A and B** for storing operands
- **3 status flags** can be used to implement conditional instructions:
  - E = 1 if ALU outputs 0xFFFFFFFF (-1)
  - $\bar{C}$ : complemented carry bit
  - N: sign bit (bit 31)

# ULg01: data path

The data path includes:

- **ALU circuit: Arithmetic Logic Unit**
- **Static memory (DRAM):** similar to your computer's processor registers
- **Dynamic memory (SRAM):** similar to your computer's RAM



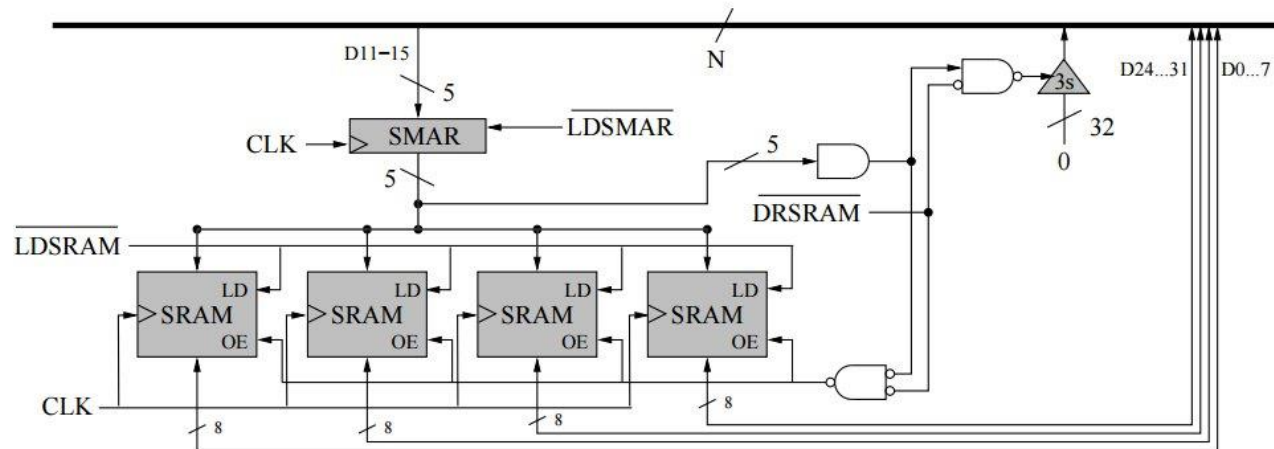
## Dynamic RAM:

- Stores programs' instructions and data
- 4 Mbytes of memory
- 1 Mword (need only 20 bits to address)

# ULg01: data path

The data path includes:

- **ALU circuit: Arithmetic Logic Unit**
- **Static memory (DRAM):** similar to your computer's processor registers
- **Dynamic memory (SRAM):** similar to your computer's RAM (Random Access Memory)

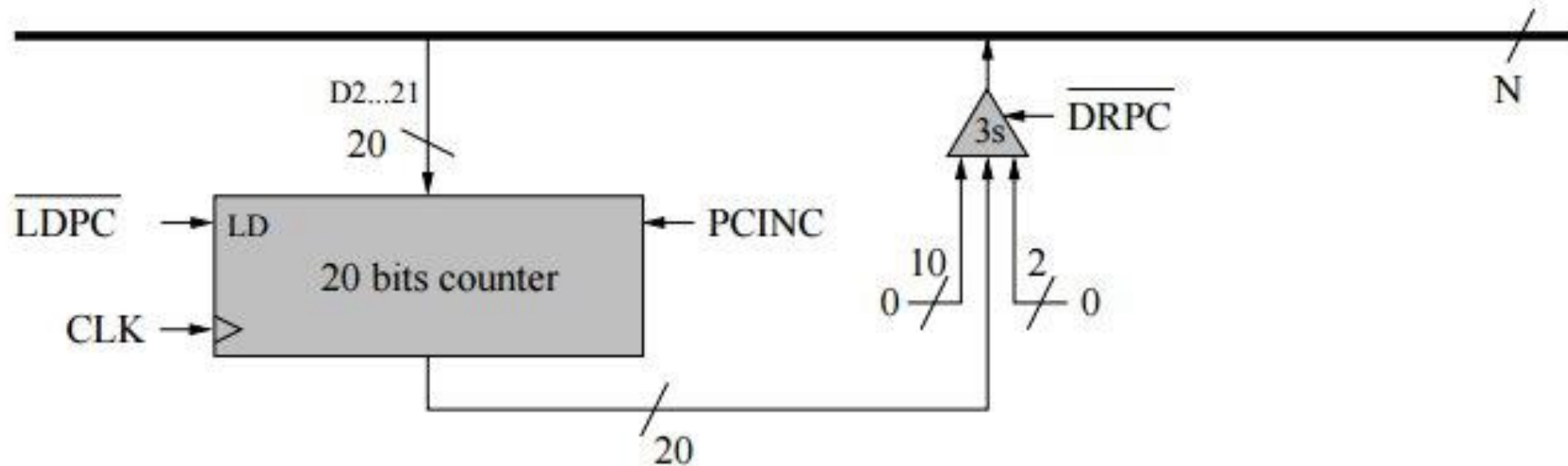


## Static RAM:

- 32 registers: R0, ..., R31
- Access to R31 always returns 0
- **More « convenient » and faster than DRAM** for data you need to access often in your program

# ULg01: program counter (PC)

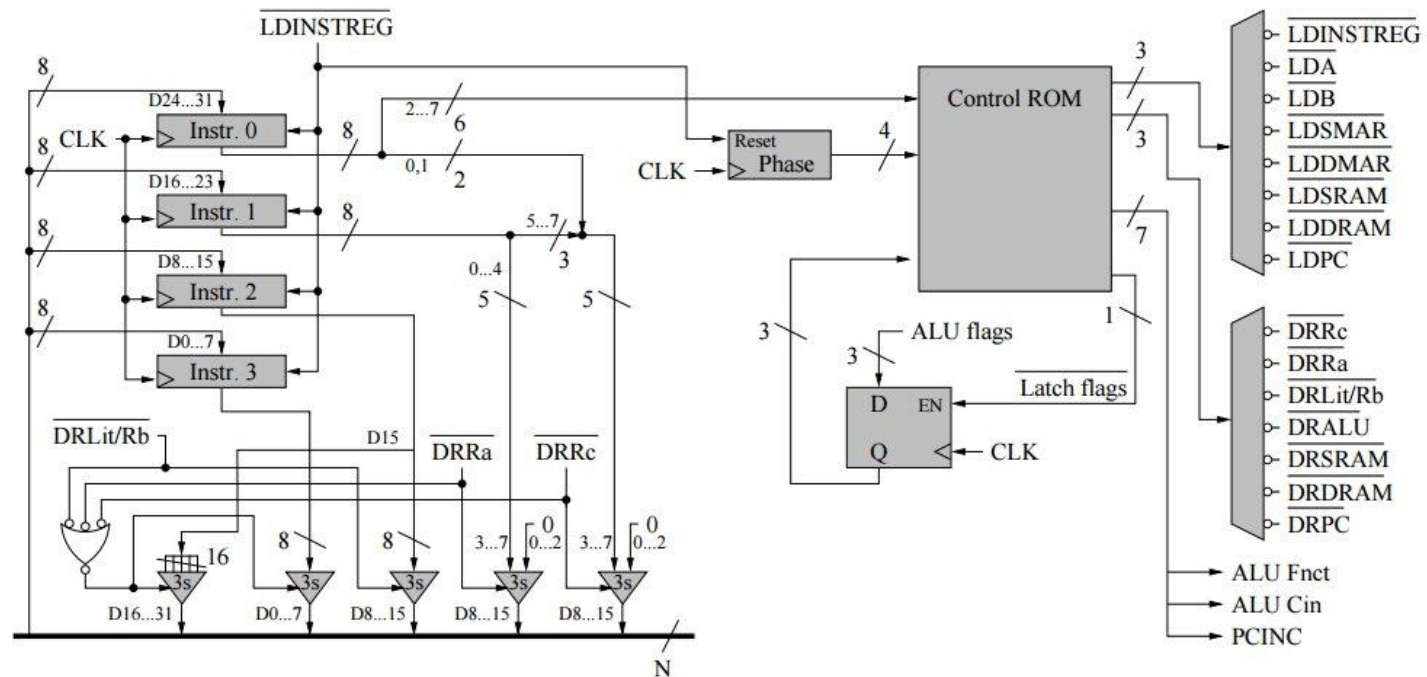
Register containing the address of the next instruction to execute





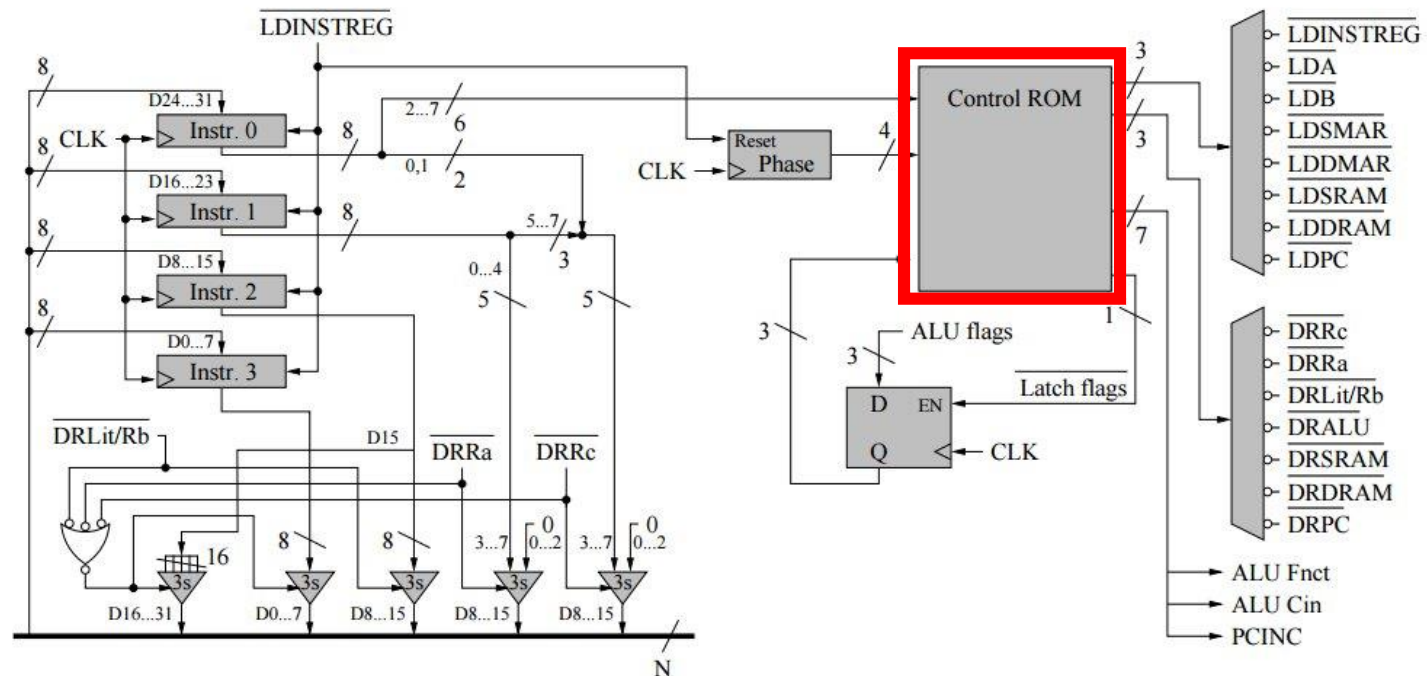
# ULg01: control unit

- The « brain » of ULg01 !



# ULg01: control unit

- The « brain » of ULg01 !
- The control unit is microprogrammed and the  $\mu$ -code is stored in a ROM.
- The  $\mu$ -code defines the behavior of ULg01 for a given 32-bits instruction.



# ULg01: microcode of OR(Ra, Rb, Rc)

Opcode	Phase	Flags	$\overline{\text{Latch}} \text{ flags}$	ALU F, $\overline{C_{in}}$ , Mode	LD SEL	DR SEL	PC+	
101001	0000	*	1	000000	011	001	0	SMAR $\leftarrow$ Ra
101001	0001	*	1	000000	001	100	0	A $\leftarrow$ SRAM
101001	0010	*	1	000000	011	010	0	SMAR $\leftarrow$ Rb
101001	0011	*	1	000000	010	100	0	B $\leftarrow$ SRAM
101001	0100	*	1	000000	011	000	0	SMAR $\leftarrow$ Rc
101001	0101	*	1	111001	101	011	0	SRAM $\leftarrow$ A   B
101001	0110	*	1	000000	100	110	1	DMAR $\leftarrow$ PC; PC+
101001	0111	*	1	000000	000	101	0	INSTREG $\leftarrow$ DRAM

- ROM address: concatenation of **Opcode**, **Phase** and **ALU flags**
- Given an address, the control ROM provides many signals that drives ULg01: LDxxxx (load), DRxxxx (drive), Latch flags, PC+,...
- **IMPORTANT**: maximum 16 phases allowed per instruction

# ULg01: microcode of OR(Ra, Rb, Rc)

Opcode	Phase	Flags	$\overline{\text{Latch}}$ flags	ALU F, $\overline{C_{in}}$ , Mode	LD SEL	DR SEL	PC+	
101001	0000	*	1	000000	011	001	0	SMAR $\leftarrow$ Ra
101001	0001	*	1	000000	001	100	0	A $\leftarrow$ SRAM
101001	0010	*	1	000000	011	010	0	SMAR $\leftarrow$ Rb
101001	0011	*	1	000000	010	100	0	B $\leftarrow$ SRAM
101001	0100	*	1	000000	011	000	0	SMAR $\leftarrow$ Rc
101001	0101	*	1	111001	101	011	0	SRAM $\leftarrow$ A   B
101001	0110	*	1	000000	100	110	1	DMAR $\leftarrow$ PC; PC+
101001	0111	*	1	000000	000	101	0	INSTREG $\leftarrow$ DRAM

ROM inputs

ROM outputs

Human-readable  
representation

# Representation of microcode for the tutorials

Ex 1	OR			
Cond	ALU	DR	LD	Sig
		Ra	SMAR	
		SRAM	A	
		Rb	SMAR	
		SRAM	B	
		Rc	SMAR	
	A   B	ALU	SRAM	
		PC	DMAR	PCINC
		DRAM	INSTREG	

**ROM inputs**                      **ROM outputs**

# ULg01

Control Inputs							Output
F3	F2	F1	F0	C	M		
0	0	1	1	1	1	0	0
1	1	0	0	1	1	0	0xffffffff
1	1	1	1	1	1	1	A
1	0	1	0	1	1	1	B
1	1	1	1	1	0	0	A-1
0	0	0	0	0	0	0	A+1
1	0	0	1	1	0	0	A+B
0	1	1	0	0	0	0	A-B
1	0	1	1	1	1	1	A and B
1	1	1	0	1	1	1	A or B
0	1	1	0	1	1	1	A xor B
0	0	0	0	1	1	1	not A
1	1	0	0	1	0	0	A+A
0	1	1	0	1	0	0	A-B-1
1	0	0	1	x	1	1	not (A xor B)
0	1	0	0	x	1	1	not (A and B)

flags:  
 E=1  $\Leftrightarrow$  Alu out = 0xffffffff  
 C : negated carry out  
 N= most sign. Alu out: a31

