Digital Systems - Mini AVR 5

Pere Palà - Alexis López

iTIC http://itic.cat

May 2016

Adding some RAM

- Up to now, we only had space for 16 registers (r16 to r31)
- Insufficient for some applications
  - We will add 1 kByte of RAM
  - 1024 x 8 bit
  - Requires 10 address bits
- Problem: OpCode Design?
  - 6 bits left to indicate operation
  - If we had 32 kByte of RAM?
  - 1 bit left for OpCode!!

Two-cycle instructions

- First Solution
- Two-cycle instruction
  - Example : `LDS Rd,k`
  - 32-bit Opcode:
  - `1001 000d dddd 0000`
  - `kkkk kkkk kkkk kkkk`
- Requires 2 cycles
- We need a state machine to keep track of the status and act accordingly.
- Allows a direct access of 64 K positions

Indirect Addressing

- Second Solution
- Use the concatenation of two 8-bit registers
  - `X <= R27 & R26`
- Example : `LD Rd,X`
  - Load the destination register with the content of the memory position formed by the concatenation of r27 and r26.
- Example : `ST X,Rr`
  - `X <= R27 & R26`
  - Store the value of the source register into the memory position formed by the concatenation of r27 and r26.
- Allows 16-bit Opcodes
- Single cycle
- Need to prepare X with the appropriate value

Documentation of AVR instructions : LD

- LD – Load Indirect from Data Space to Register using Index X
- Description:
  - Loads one byte indirect from the data space to a register. The data location is pointed to by the X (16 bits) Pointer Register in the Register File.
- Rd $\leftarrow$ (X)
- Syntax: LD Rd, X
- Operands: $0 \leq d \leq 31$
- Program Counter: PC $\leftarrow$ PC + 1
- 16-bit Opcode: `1001 000d dddd 1100`
- Status Register (SREG) and Boolean Formula:

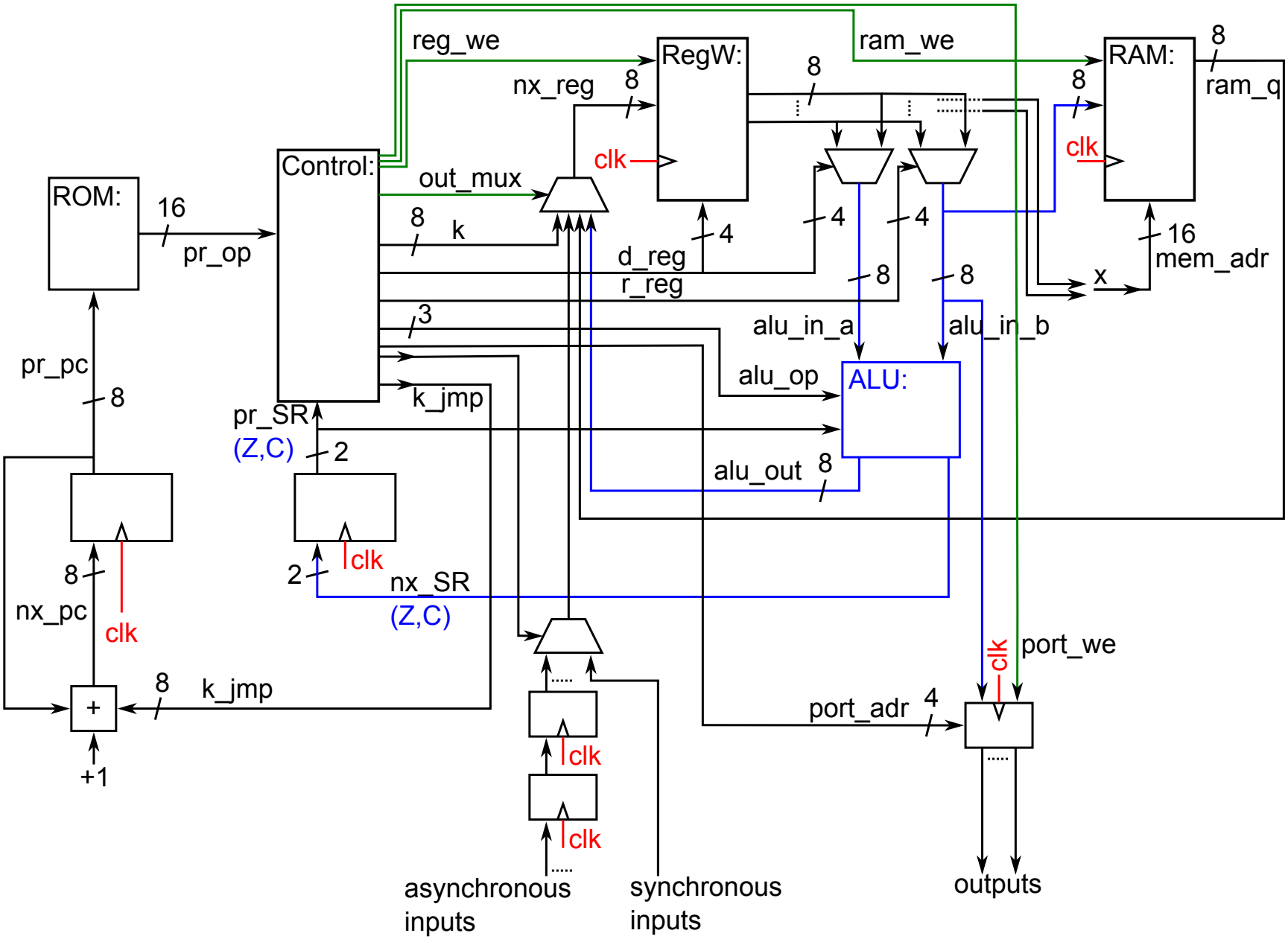
```
I T H S V N Z C
- - - - - - - -
```

Documentation of AVR instructions : ST

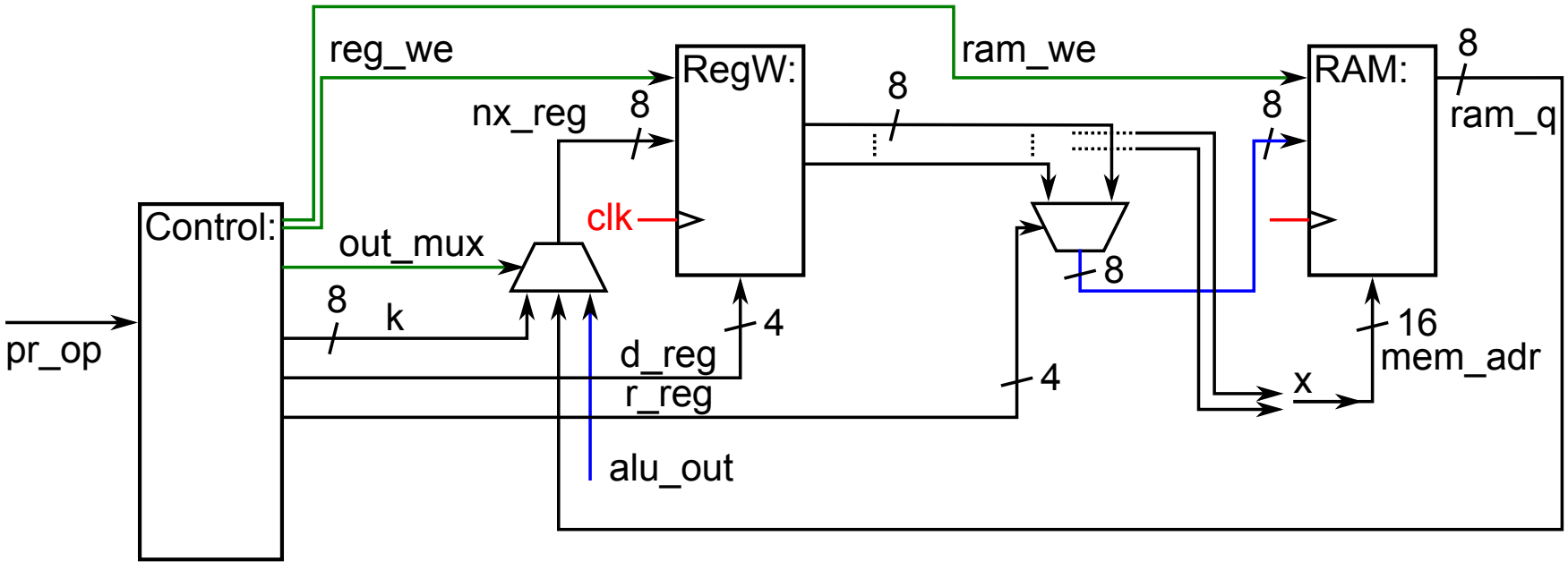
- ST – Store Indirect From Register to Data Space using Index X
- Description:
  - Stores one byte indirect from a register to data space. The data location is pointed to by the X (16 bits) Pointer Register in the Register File.
- (X) $\leftarrow$ Rr
- Syntax: ST X, Rr
- Operands: $0 \leq r \leq 31$
- Program Counter: PC $\leftarrow$ PC + 1
- 16-bit Opcode: `1001 001r rrrr 1100`
- Status Register (SREG) and Boolean Formula:

```
I T H S V N Z C
- - - - - - - -
```

Overall System View



Detailed View of RAM portion



Sample Code

```
NOP
LDI   r26,x05
LDI   r27,x00  ; x=0005;
LDI   r16,x01  ;
LDI   r18,xFF  ; do something
ST    X,r16    :
LD    r17,X    ;
RJMP  -1       ; HALT
```

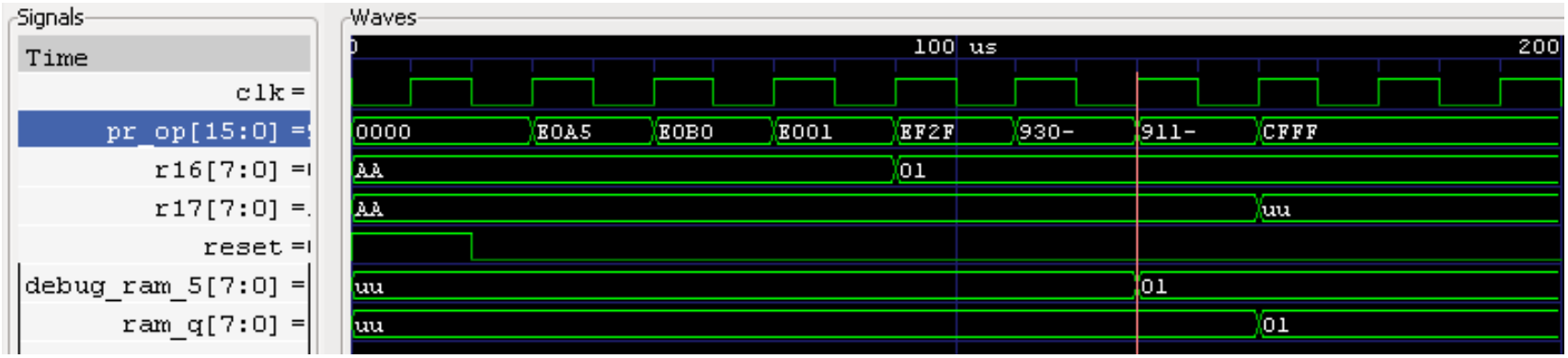
VHDL implementation. RAM

```
constant RAM_size : integer := 2 ** 10 - 1;    -- 1 kByte RAM
type RAM_block is array (0 to RAM_size) of
                                    std_logic_vector(7 downto 0);
   signal RAM     : RAM_block;
   signal ram_q   : std_logic_vector( 7 downto 0);
   signal mem_adr : std_logic_vector(15 downto 0);
```

VHDL implementation. RAM with Synchronous Read and Write

- Option 1 (read before write):

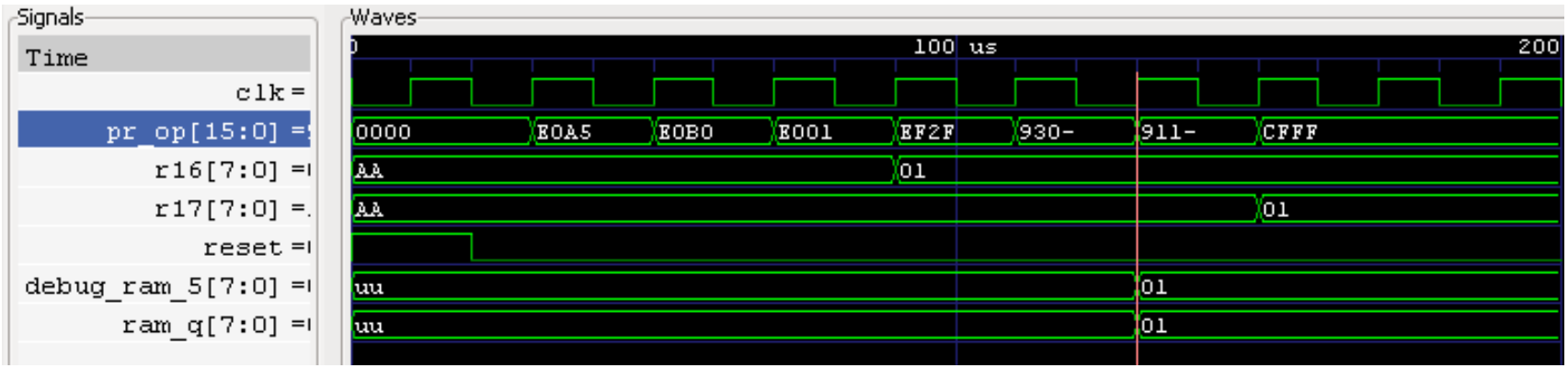
```
RAM_process : process(clk)
begin
  if rising_edge(clk) then
     if mem_we ='1' then
        RAM(to_integer(unsigned(mem_adr(9 downto 0)))) <=
            regs(to_integer(unsigned(r_reg)));
     end if;
     ram_q <= RAM(to_integer(unsigned(mem_adr(9 downto 0))));
  end if;
end process;
```



VHDL implementation. Ram with Synchronous Write and "Combinational" Read

- Option 2 (write before read):

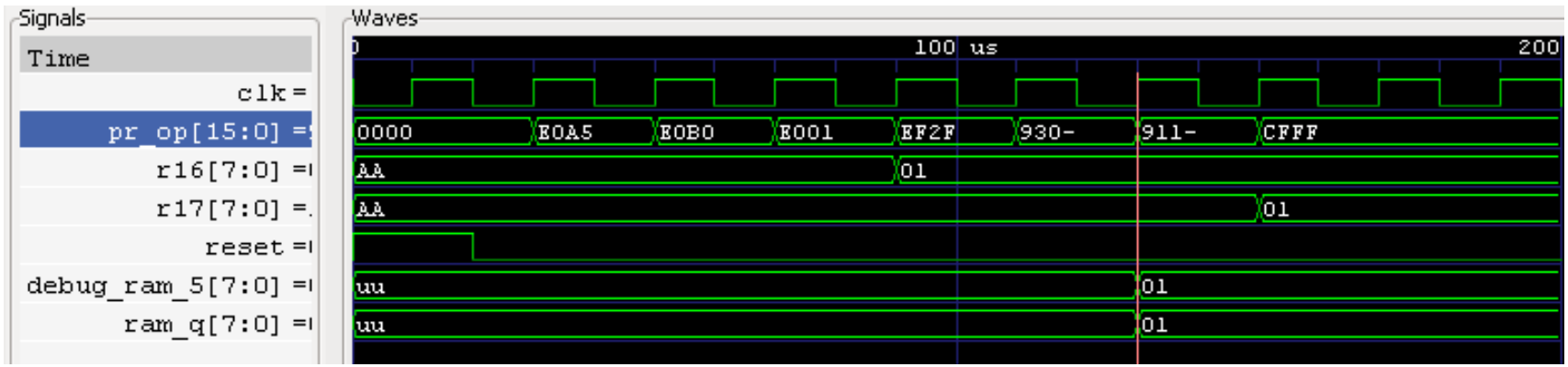
```
RAM_process : process(clk)
begin
  if rising_edge(clk) then
     if mem_we ='1' then
        RAM(to_integer(unsigned(mem_adr(9 downto 0)))) <=
            regs(to_integer(unsigned(r_reg)));
     end if;
  end if;
end process;
ram_q <= RAM(to_integer(unsigned(mem_adr(9 downto 0))));
```



VHDL implementation. Ram with Synchronous Write and Combinational Read. Synthesizes good

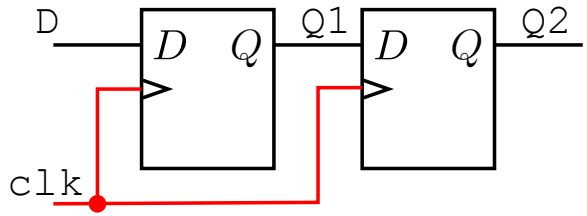
- Option 3: (write before read with registered address)

```
RAM_process : process(clk)
begin
   if rising_edge(clk) then
      if mem_we ='1' then
         RAM(to_integer(unsigned(mem_adr(9 downto 0)))) <=
             regs(to_integer(unsigned(r_reg)));
      end if;
      mem_read_adr <= mem_adr;
   end if;
end process;
ram_q <= RAM(to_integer(unsigned(mem_read_adr(9 downto 0))));
```



On the importance of coding style and available features

- Option 1: Read before write
  - Not desirable!
- Why this behavior?
- VHDL code for:




```
process(clk)
begin
   if rising_edge(clk) then
      Q1 <= D;
      Q2 <= Q1;
   end if;
end process;
```

- Which is exactly the style in Option 1

On the importance of coding style and available features

- Option 2:
  - Compilation time : > 5 min

Total logic elements	10,717 / 22,320 (48 %)
Total combinational functions	6,629 / 22,320 (30 %)
Dedicated logic registers	8,236 / 22,320 (37 %)
Total registers	8236
Total pins	9 / 154 (6 %)
Total virtual pins	0
Total memory bits	0 / 608,256 (0 %)
Embedded Multiplier 9-bit elements	0 / 132 (0 %)
Total PLLs	1 / 4 (25 %)

- Option 3:
  - Compilation time : < 1 min

Total logic elements	124 / 22,320 (< 1 %)
Total combinational functions	124 / 22,320 (< 1 %)
Dedicated logic registers	53 / 22,320 (< 1 %)
Total registers	53
Total pins	9 / 154 (6 %)
Total virtual pins	0
Total memory bits	8,192 / 608,256 (1 %)
Embedded Multiplier 9-bit elements	0 / 132 (0 %)
Total PLLs	1 / 4 (25 %)