CSU22022 Computer Architecture I

Twentieth Lecture - Instruction Set and Memory

Michael Manzke

2023-2024

Trinity College Dublin

Basic Computer Architecture

- Computers consist of:
  - Datapath
  - Control unit
- It is designed to implement a particular instruction set
- The individual instructions are the engineering equivalent of the mathematician's:
  - $Z = f(x, y)$

Opcode – Destination - Operands

- OPCODE
  - Selects the function
- DESTINATION
  - Is nearly always a datapath register
- OPERANDS
  - Usually come from datapath register

Instruction Format Examples

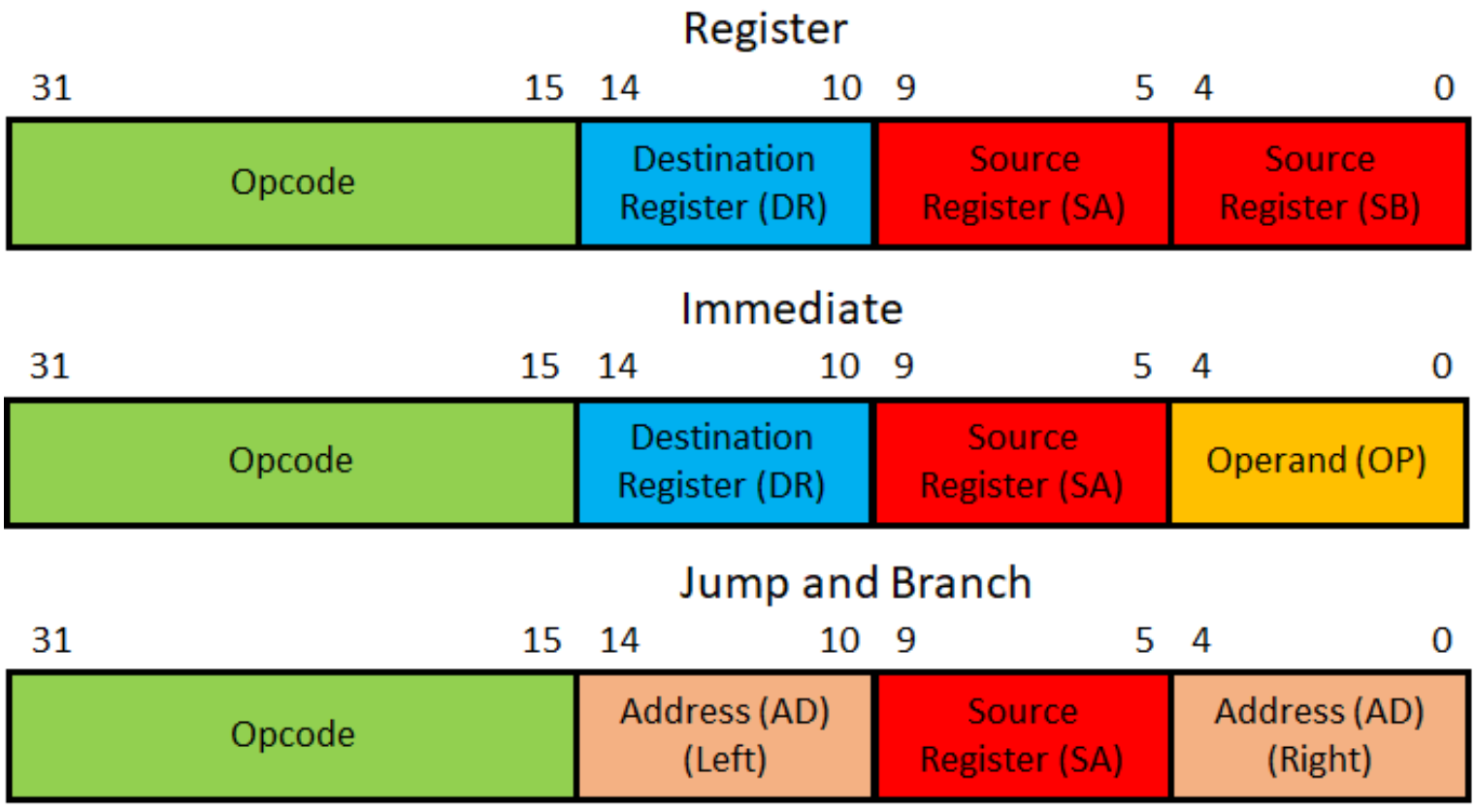

Figure 1: How we use DR, SA, and SB "payload" is determined by the selected ASM in the control memory. The opcode selects the ASM.

Register Instruction

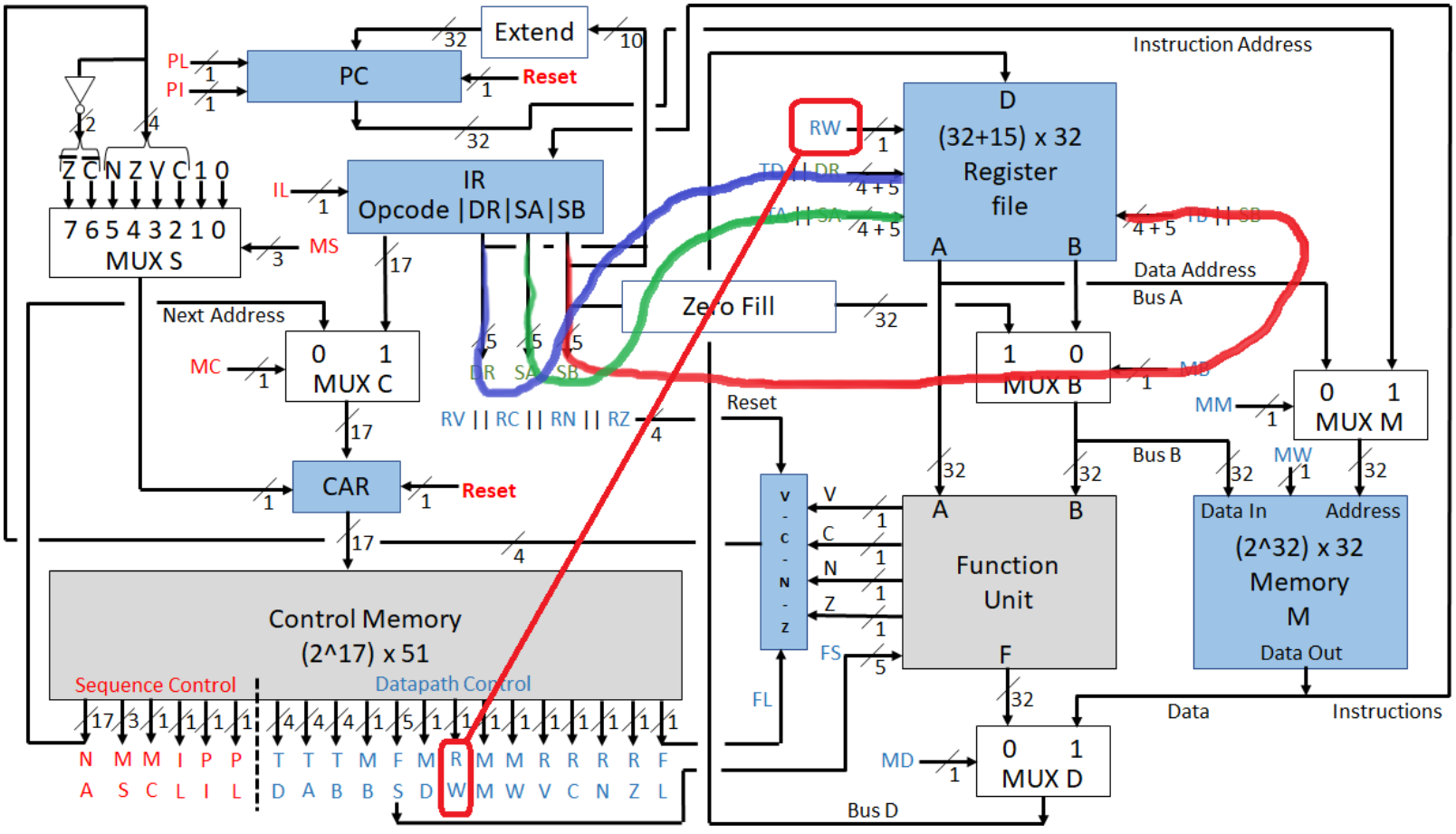

Figure 2: The RW signal determines whether we use the the DR, SA, and SB signals to select registers.

Immediate Instruction

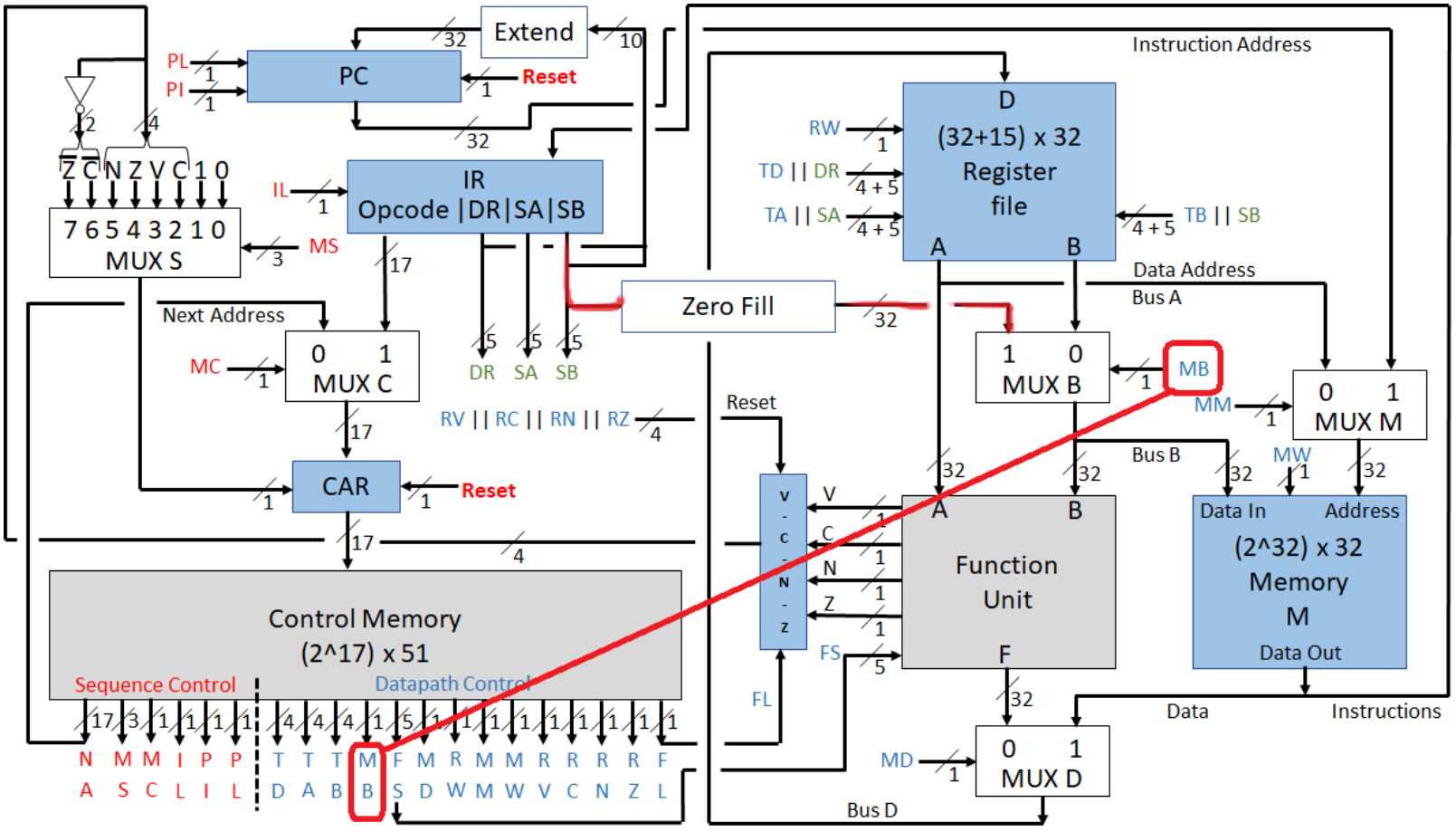

Figure 3: MB signal determines whether we use SB as an immediate value or use SB to select a source register for the B port of the regsiter file.

Jump and Branch Instruction

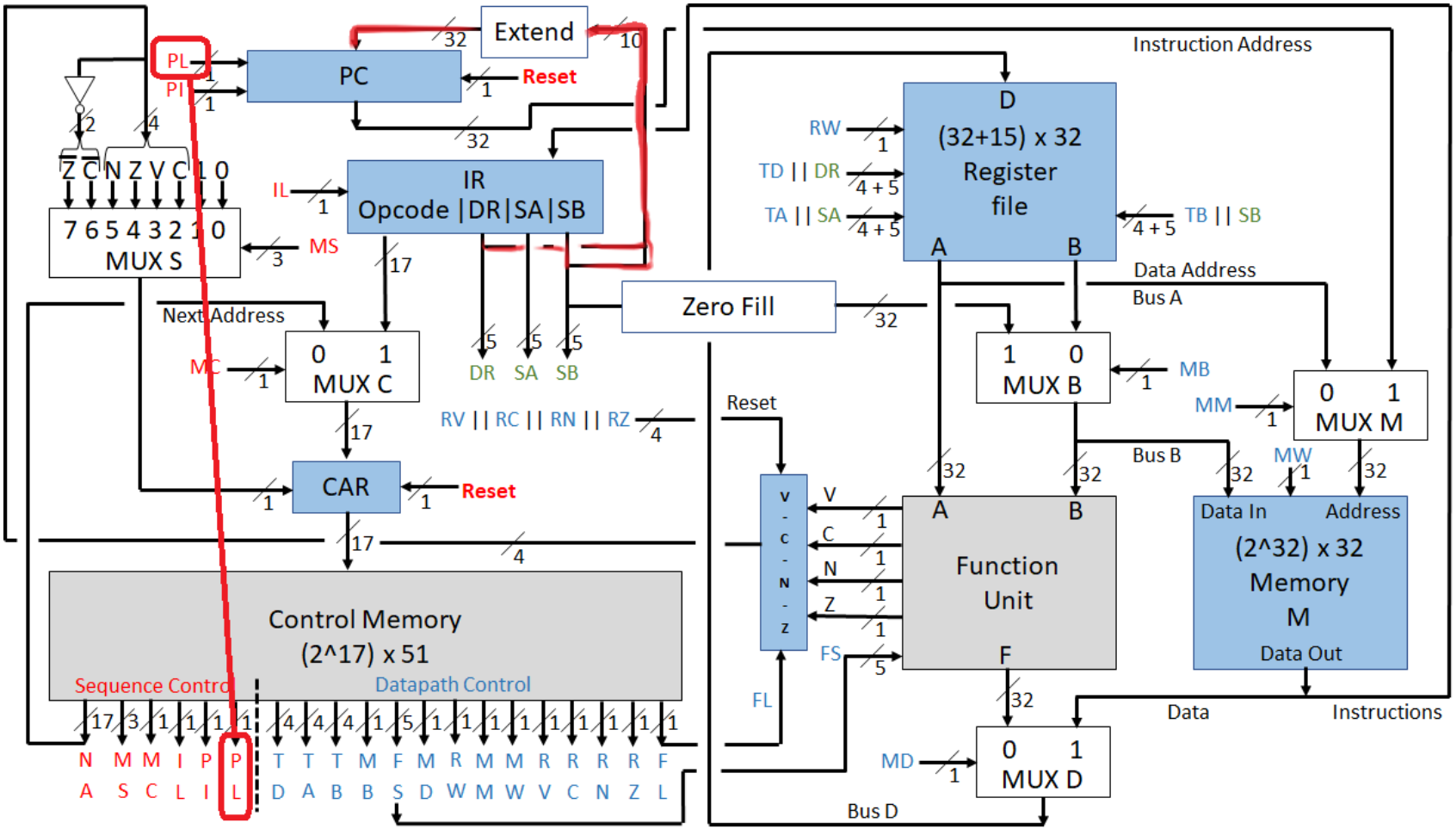

Figure 4: The PL signal determines if we use use DR and SB as a 2's complement number that could be added to the program Counter (PC).

Random Access Memory (RAM) Implementation

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.NUMERIC_STD.ALL;
4
5  entity CPU_RAM_XXXXXXXX is
6      Port ( Clock : in STD_LOGIC;
7             Address : in STD_LOGIC_VECTOR (31 downto 0);
8             DataIn : in STD_LOGIC_VECTOR (31 downto 0);
9             WriteEnable : in STD_LOGIC;
10            DataOut : out STD_LOGIC_VECTOR (31 downto 0));
11 end CPU_RAM_XXXXXXXX;
12
13 architecture Behavioral of CPU_RAM_XXXXXXXX is
14
15 -- we use the least significant 7 bit of the address
16 type RAM_array is array(0 to 127) of STD_LOGIC_VECTOR (31
       downto 0);
```

Listing 1: CPU_RAM_XXXXXXXX entity

Random Access Memory (RAM) Implementation

```
signal RAM : RAM_array:=(
X"00000000",-- 00
X"00000001",-- 01
X"00000002",-- 02
X"00000003",-- 03
-- Machine code
-- example studentID 87654321
-- your machine code starts at digit 3 of your ID = 4
-- Opcode = digit 3 = 4
-- DR = digit 2 = 3
-- SA = digit 1 = 2
-- SB = digit 0 = 1
--        Opcode              DR        SA        SB
  "0000000000000100"&"00011"&"00010"&"00001",-- 04
  "0000000000000101"&"00100"&"00011"&"00010",-- 05
  "0000000000000110"&"00101"&"00100"&"00011",-- 06
  "0000000000000111"&"00110"&"00101"&"00100",-- 07
  "0000000000001000"&"00111"&"00110"&"00101",-- 08
```

Listing 2: Initialisation of the RAM_array with data and instructions.

Random Access Memory (RAM) Implementation

```
1  X"0000007F" -- 7F
2  );
3  begin
4  process (Clock)
5  begin
6     if Clock'event and Clock='1' then
7       if WriteEnable='1' then
8         RAM(to_integer(unsigned(Address(6 downto 0)))) <=
       DataIn after 2ns;
9       end if;
10    end if;
11 end process;
12
13 DataOut <= RAM(to_integer(unsigned(Address(6 downto 0))))
       after 2ns;
14
15 end Behavioral;
```

Listing 3: Initialisation continous to address $7F_{16} or 127_{10}$. We only use 7 of the 32 address bits.

Transfer data from the RAM to the Register File

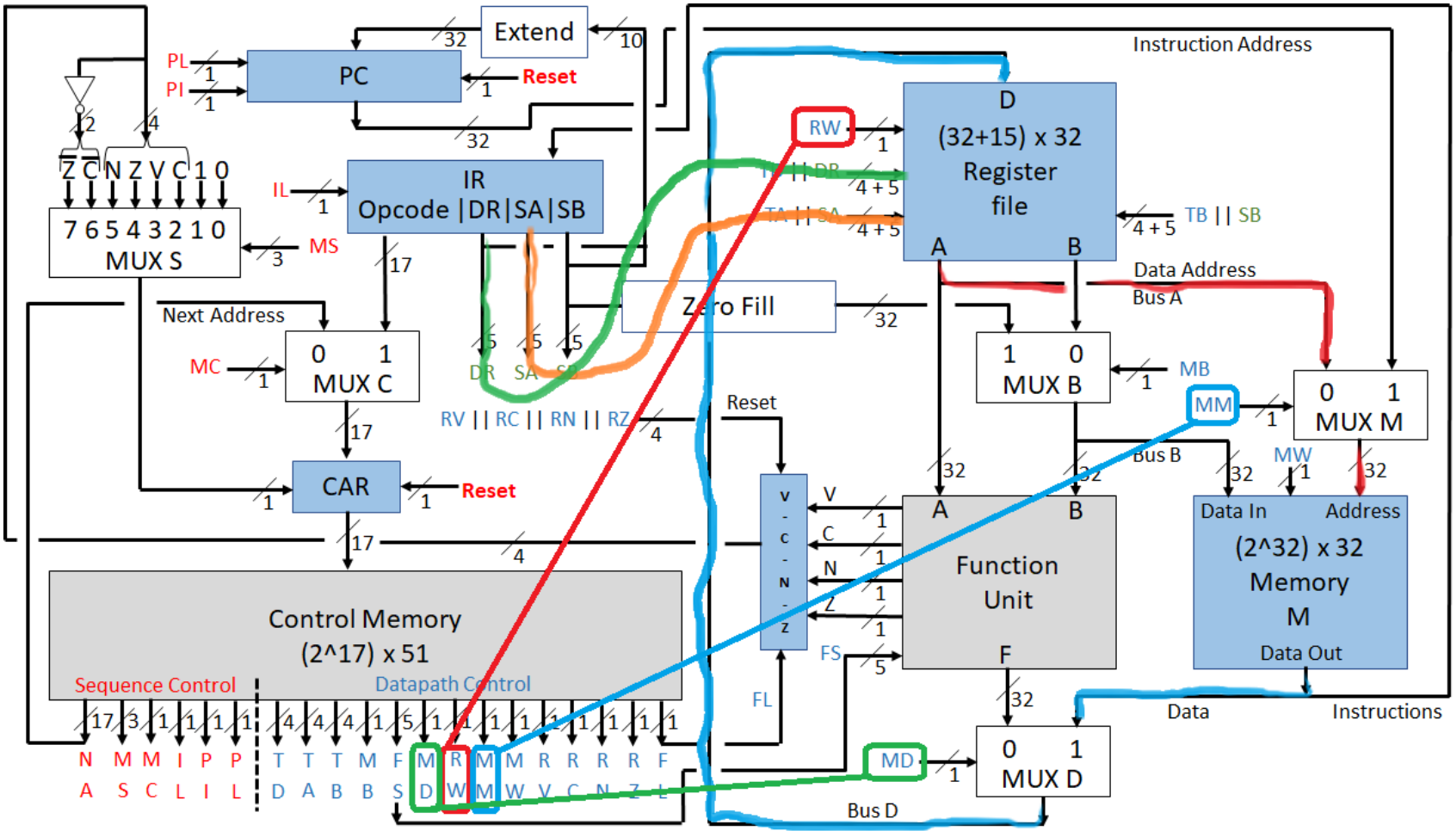

Figure 5: Load operation

Transfer data from the Register File to the RAM

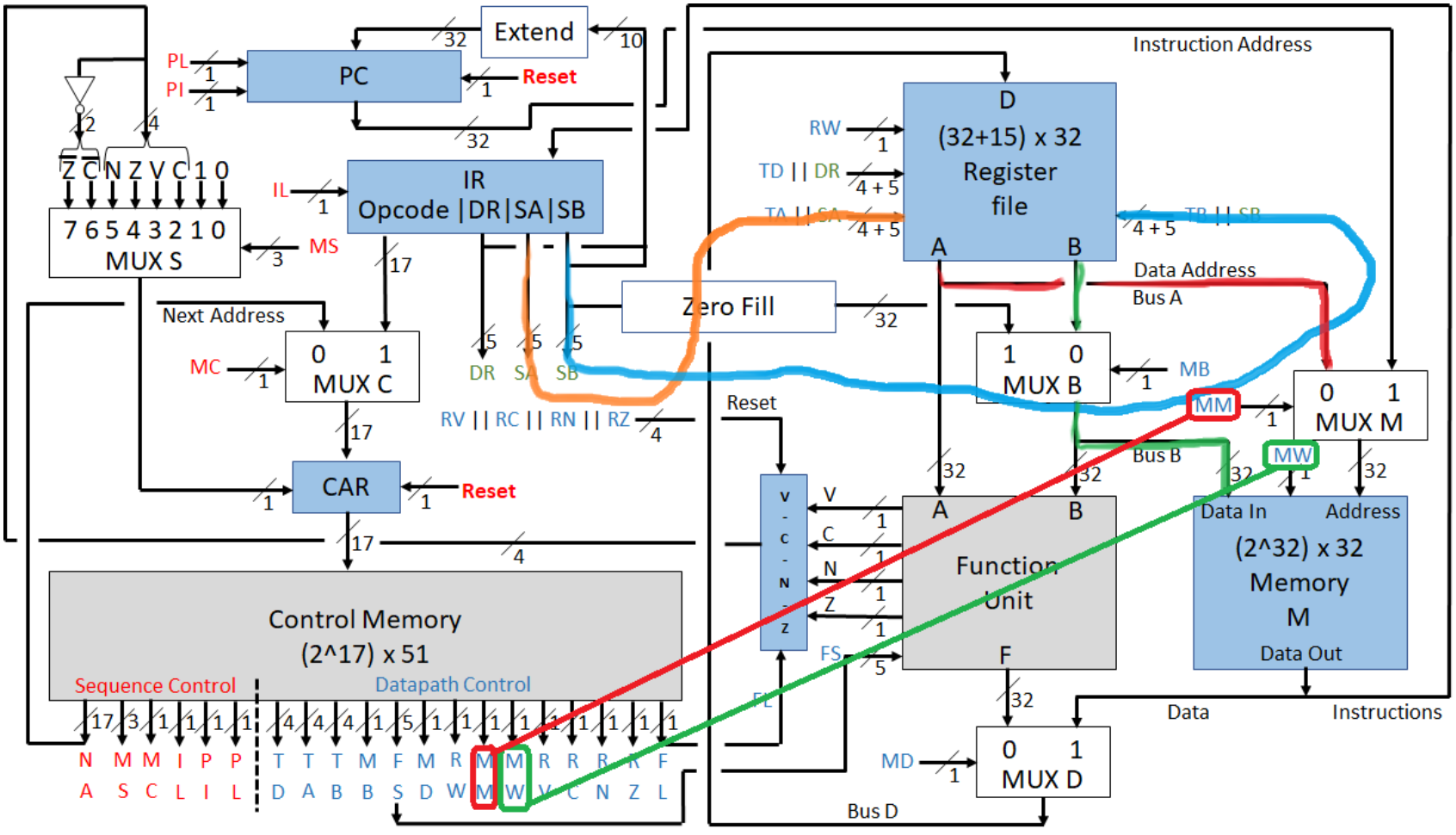

Figure 6: Store operation

Instruction Fetch $IR \leftarrow M[PC]$

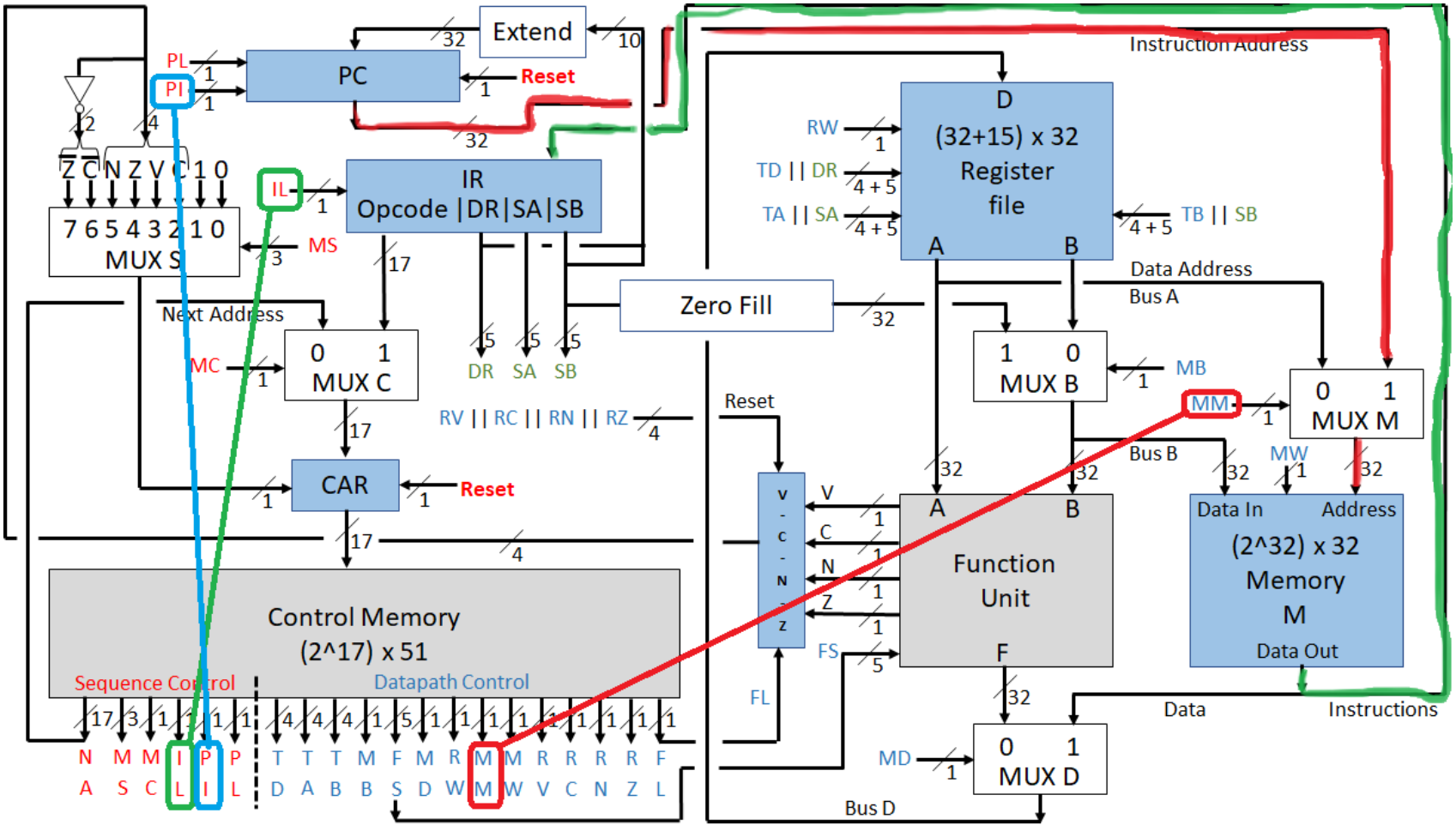

Figure 7: PC points into the RAM to the next to be executed instruction.

Program Counter (PC) Implementation

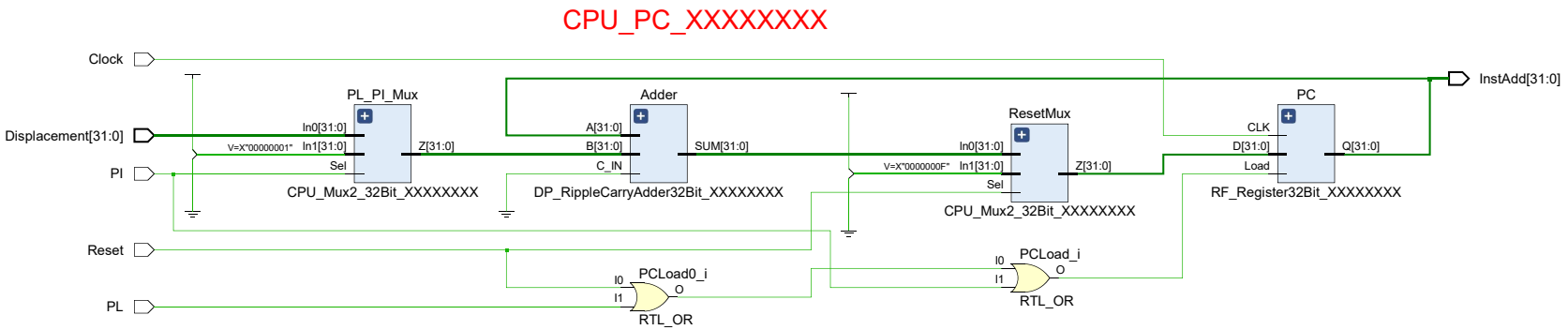

Figure 8: PC points into the RAM to the next to be executed instruction. If we execute an Instruction Fetch (IL) operation, will also execute a PC increment (PI) operation. See Figure 8.

Instructions in the RAM

```
1  -- Machine code
2  -- example studentID 87654321
3  -- your machine code starts at digit 3 of your ID = 4
4  -- Opcode = digit 3 = 4
5  -- DR = digit 2 = 3
6  -- SA = digit 1 = 2
7  -- SB = digit 0 = 1
8  --        Opcode             DR        SA        SB
9    "0000000000000100"&"00011"&"00010"&"00001",-- 04
10   "0000000000000101"&"00100"&"00011"&"00010",-- 05
11   "0000000000000110"&"00101"&"00100"&"00011",-- 06
12   "0000000000000111"&"00110"&"00101"&"00100",-- 07
13   "0000000000001000"&"00111"&"00110"&"00101",-- 08
```

Listing 4: Initialisation of the RAM_array with instructions. Memory addresses 04_{16} to 08_{16}. If the PC points to the instruction at address 05_{16} we will transfer that instruction into the Instruction Register IR and increment the PC. Therefore the PC will point to the next instruction at address 06_{16}.