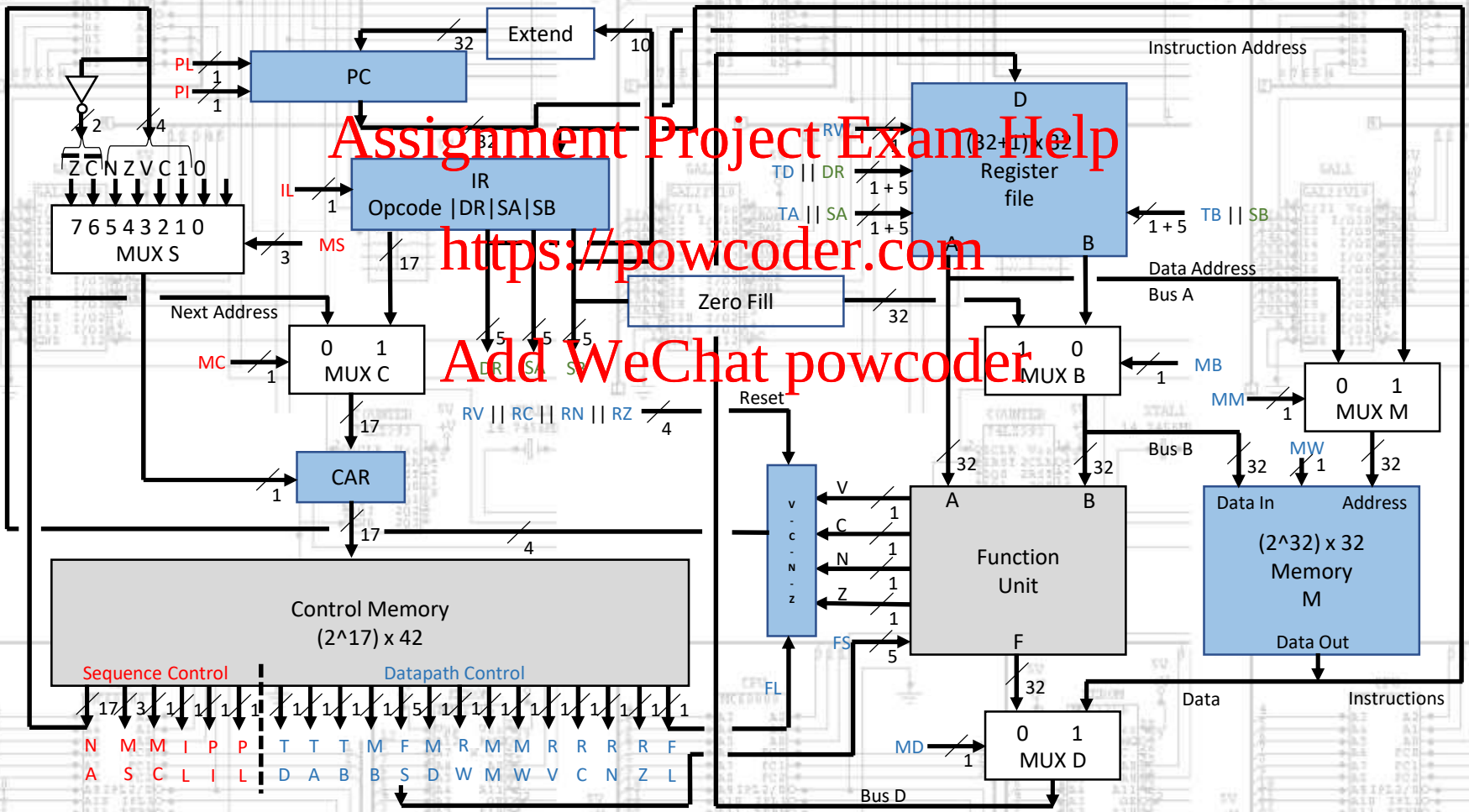
Multiple-Cycle Microprogrammed Computer



Project 2
Microcoded Instruction Set Processor

- Project 2 in incremental steps
- modifications are required:
  - Increase the number of registers in the register-file from 32 to 33
  - This requires an additional select bit for the two multiplexers (Bus A and Bus B) and the destination decoder. These are separate signals (TD, TA, TB) that are provided by the Control Memory
  - The size of the registers in the register-file has to be 32 bit (size of instructions)

Datapath Modifications

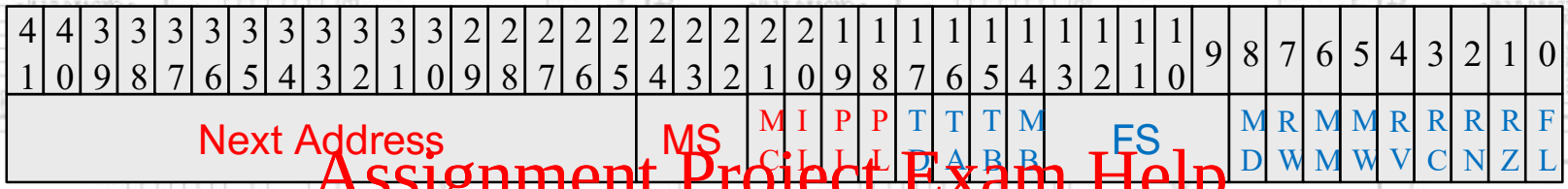
- Consequently, all components of the Datapath:
  - MUXs in the Register file
  - Decoder in the Register file
  - Arithmetic/logic Unit
  - Shifter and MUXs …
- are 32 bit

Datapath Modifications

- Add and test:
  - Memory M (512 x 32)
  - Control Memory (256 x 42)
- to your project.
  - MUX M will feed 32 bit addresses from ether the Bus A or the PC into the Memory M entity but only the 9 least significant address bits will be used to index into the array. This restricts the memory size to 512.

Control Memory 256 x 42

library IEEE




```
-- Michael Manzke
-- michael.manzke@cs.tcd.ie
-- 3rd December 2020

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
```


```
entity control_memory is
    Port (FL : out std_logic;                                  -- 0
          RZ : out std_logic;                                  -- 1
          RN : out std_logic;                                  -- 2
          RC : out std_logic;                                  -- 3
          RV : out std_logic;                                  -- 4
          MW : out std_logic;                                  -- 5
          MM : out std_logic;                                  -- 6
          RW : out std_logic;                                  -- 7
          MD : out std_logic;                                  -- 8
          FS : out std_logic_vector(4 downto 0);               -- 9 to 13
          MB : out std_logic;                                  -- 14
          TB : out std_logic;                                  -- 15
          TA : out std_logic;                                  -- 16
          TD : out std_logic;                                  -- 17
          PL : out std_logic;                                  -- 18
          PI : out std_logic;                                  -- 19
          IL : out std_logic;                                  -- 20
          MC : out std_logic;                                  -- 21
          MS : out std_logic_vector(2 downto 0);               -- 22 to 24
          NA : out std_logic_vector(16 downto 0);              -- 25 to 41
          IN_CAR : in std_logic_vector(16 downto 0));
end control_memory;
```


```
architecture Behavioral of control_memory is
type mem_array is array(0 to 255) of std_logic_vector(41 downto 0);

begin

memory_m: process(IN_CAR)
        variable control_mem : mem_array:=(
-- |41            25|2422|21|20|19|18|17|16|15|14|13  9|8|7|6|5|4|3|2|1|0|
-- | Next Address   | MS |MC|IL|PI|PL|TD|TA|TB|MB|   FS |MD|RW|MM|MW|RV|RC|RN|RZ|FL|
-- | Next Address   | MS | C| L| I| L| D| A| B| B|   FS |D|W|M|W|V|C|N|Z|L|
   "00000000000000000 000  0  0  0  0  0  0  0  0 00000 0 0 0 0 0 0 0 0 0",-- 00
   "00000000000000000 000  0  0  0  0  0  0  0  0 00000 0 0 0 0 0 0 0 0 0",-- 01
   "00000000000000000 000  0  0  0  0  0  0  0  0 00000 0 0 0 0 0 0 0 0 0",-- 02
   "00000000000000000 000  0  0  0  0  0  0  0  0 00000 0 0 0 0 0 0 0 0 0",-- 03
   "00000000000000000 000  0  0  0  0  0  0  0  0 00000 0 0 0 0 0 0 0 0 0",-- 04
   "00000000000000000 000  0  0  0  0  0  0  0  0 00000 0 0 0 0 0 0 0 0 0",-- 05
   "00000000000000000 000  0  0  0  0  0  0  0  0 00000 0 0 0 0 0 0 0 0 0",-- 06
   "00000000000000000 000  0  0  0  0  0  0  0  0 00000 0 0 0 0 0 0 0 0 0",-- 07
```

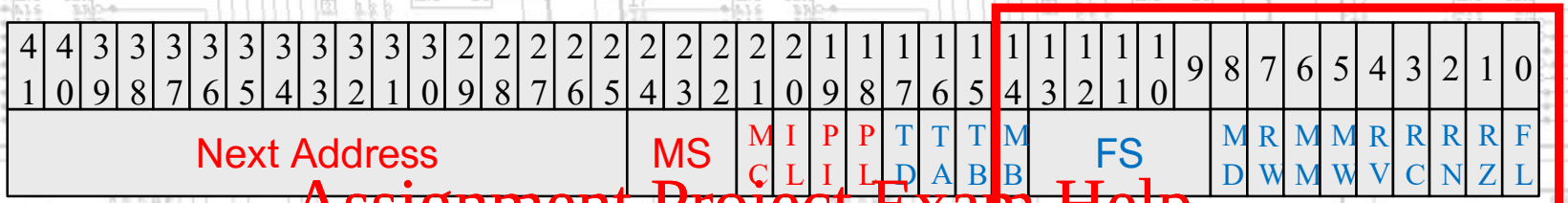
-- Address $08 to $17

```
-- |41          25|2422|21|20|19|18|17|16|15|14|13  9|8|7|6|5|4|3|2|1|0|
-- | Next Address  | MS | M| I| P| P| T| T| T| M|  FS |M|R|M|M|R|R|R|R|F|
-- | Next Address  | MS | C| L| I| L| D| A| B| B|  FS |D|W|M|W|V|C|N|Z|L|
  "0000000000000000 000  0  0  0  0  0  0  0  0 00000 0 0 0 0 0 0 0 0 0",-- 08
  "0000000000000000 000  0  0  0  0  0  0  0  0 00000 0 0 0 0 0 0 0 0 0",-- 09
  "0000000000000000 000  0  0  0  0  0  0  0  0 00000 0 0 0 0 0 0 0 0 0",-- 0A
  "0000000000000000 000  0  0  0  0  0  0  0  0 00000 0 0 0 0 0 0 0 0 0",-- 0B
  "0000000000000000 000  0  0  0  0  0  0  0  0 00000 0 0 0 0 0 0 0 0 0",-- 0C
  "0000000000000000 000  0  0  0  0  0  0  0  0 00000 0 0 0 0 0 0 0 0 0",-- 0D
  "0000000000000000 000  0  0  0  0  0  0  0  0 00000 0 0 0 0 0 0 0 0 0",-- 0E
  "0000000000000000 000  0  0  0  0  0  0  0  0 00000 0 0 0 0 0 0 0 0 0",-- 0F
-- |41          25|2422|21|20|19|18|17|16|15|14|13  9|8|7|6|5|4|3|2|1|0|
-- | Next Address  | MS | M| I| P| P| T| T| T| M|  FS |M|R|M|M|R|R|R|R|F|
-- | Next Address  | MS | C| L| I| L| D| A| B| B|  FS |D|W|M|W|V|C|N|Z|L|
  "0000000000000000 000  0  0  0  0  0  0  0  0 00000 0 0 0 0 0 0 0 0 0",-- 10
  "0000000000000000 000  0  0  0  0  0  0  0  0 00000 0 0 0 0 0 0 0 0 0",-- 11
  "0000000000000000 000  0  0  0  0  0  0  0  0 00000 0 0 0 0 0 0 0 0 0",-- 12
  "0000000000000000 000  0  0  0  0  0  0  0  0 00000 0 0 0 0 0 0 0 0 0",-- 13
  "0000000000000000 000  0  0  0  0  0  0  0  0 00000 0 0 0 0 0 0 0 0 0",-- 14
  "0000000000000000 000  0  0  0  0  0  0  0  0 00000 0 0 0 0 0 0 0 0 0",-- 15
  "0000000000000000 000  0  0  0  0  0  0  0  0 00000 0 0 0 0 0 0 0 0 0",-- 16
  "0000000000000000 000  0  0  0  0  0  0  0  0 00000 0 0 0 0 0 0 0 0 0",-- 17
```

-- Address $F8 to $FF

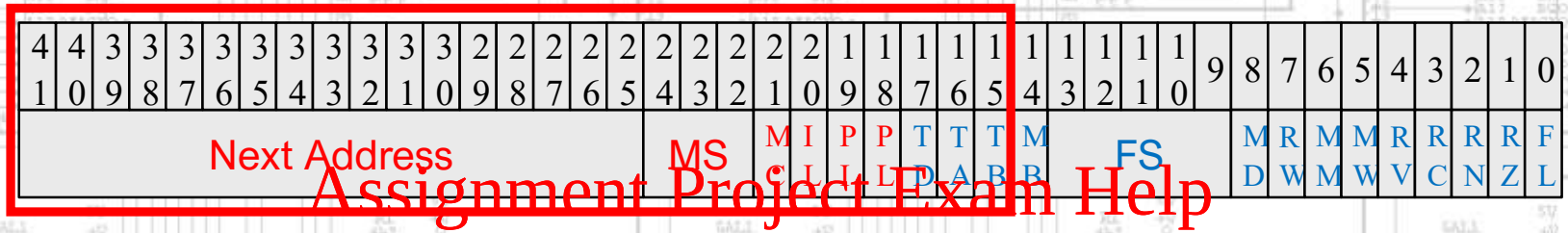

```
                                                                    :
                                                                    :
                                                                    :
-- |41           25|2422|21|20|19|18|17|16|15|14|13  9|8|7|6|5|4|3|2|1|0|
-- | Next Address  | MS | M| I| P| P| T| T| T| M|  FS |M|R|M|M|R|R|R|R|F|
-- | Next Address  | MS | C| L| I| L| D| A| B| S|  FS |D|W|M|W|V|C|N|Z|L|
   "0000000000000000 000  0  0  0  0  0  0  0  0 00000 0 0 0 0 0 0 0 0 0",-- F8
   "0000000000000000 000  0  0  0  0  0  0  0  0 00000 0 0 0 0 0 0 0 0 0",-- F9
   "0000000000000000 000  0  0  0  0  0  0  0  0 00000 0 0 0 0 0 0 0 0 0",-- FA
   "0000000000000000 000  0  0  0  0  0  0  0  0 00000 0 0 0 0 0 0 0 0 0",-- FB
   "0000000000000000 000  0  0  0  0  0  0  0  0 00000 0 0 0 0 0 0 0 0 0",-- FC
   "0000000000000000 000  0  0  0  0  0  0  0  0 00000 0 0 0 0 0 0 0 0 0",-- FD
   "0000000000000000 000  0  0  0  0  0  0  0  0 00000 0 0 0 0 0 0 0 0 0",-- FE
   "0000000000000000 000  0  0  0  0  0  0  0  0 00000 0 0 0 0 0 0 0 0 0",-- FF
                                        );
        variable addr : integer;
        variable control_out : std_logic_vector(41 downto 0);
```

Begin (process) LSB




```
begin
        addr := conv_integer(IN_CAR);
        control_out := control_mem(addr);
        FL <= control_out(0);
        RZ <= control_out(1);
        RN <= control_out(2);
        RC <= control_out(3);
        RV <= control_out(4);
        MW <= control_out(5);
        MM <= control_out(6);
        RW <= control_out(7);
        MD <= control_out(8);
        FS <= control_out(13 downto 9);
        MB <= control_out(14);
```

Begin (process) MSB




```
        TB <= control_out(15);
        TA <= control_out(16);
        TD <= control_out(18);
        PL <= control_out(17);
        PI <= control_out(19);
        IL <= control_out(20);
        MC <= control_out(21);
        MS <= control_out(24 downto 22);
        NA <= control_out(41 downto 25);
        end process;
end Behavioral;
```

VHDL top-level models

- The Modified register-file
- The Functional Unit
- The two memories

Block Diagram