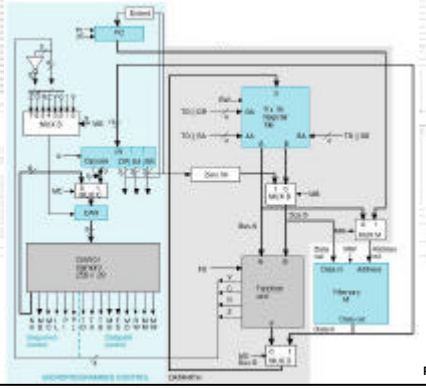
Multiple-Cycle Microprogrammed Computer



Project 2
Microcoded Instruction Set Processor

- Project 2 in incremental steps
- modifications are required for tomorrow:
  - Increase the number of registers in the register-file from 8 to 9
  - 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 increased to 16bit (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 …
- have to be adjusted to 16bit operations.

Datapath Modifications

- Add and test:
  - Memory M (512 x 16)
  - Control Memory (256 x 28)
- to your project.
  - MUX M will feed 16 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 28

library IEEE


```
-- Michael Manzke
-- michael.manzke@cs.tcd.ie
-- 25 April 2003

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


entity control_memory

```
entity control_memory is
    Port( MW : out std_logic;
          MM : out std_logic;
          RW : out std_logic;
          MD : out std_logic;
          FS : out std_logic_vector(4 downto 0);
          MB : out std_logic;
          TB : out std_logic;
          TA : out std_logic;
          TD : out std_logic;
          PL : out std_logic;
          PI : out std_logic;
          IL : out std_logic;
          MC : out std_logic;
          MS : out std_logic_vector(2 downto 0);
          NA : out std_logic_vector(7 downto 0);
          IN_CAR : in std_logic_vector(7 downto 0));
end control_memory;
```


architecture Behavioral of control_memory is

```
architecture Behavioral of control_memory is
type mem_array is array(0 to 255) of std_logic_vector(27 downto 0);

begin

memory_m: process(IN_CAR)
        variable control_mem : mem_array:=(
                                        -- 0
                                        X"FFFFFFF", -- 0
                                        X"0000000", -- 1
                                        X"AAAAAAA", -- 2
                                        X"0000000", -- 3
                                        X"BBBBBBB", -- 4
                                        X"0000000", -- 5
                                        X"CCCCCCC", -- 6
                                        X"0000000", -- 7
```


-- Address $08 to $17

```
                                        X"DDDDDDD", -- 8
                                        X"0000000", -- 9
                                        X"1111111", -- A
                                        X"0000000", -- B
                                        X"2222222", -- C
                                        X"0000000", -- D
                                        X"3333333", -- E
                                        X"0000000", -- F
                                        -- 1
                                        X"0000000", -- 0
                                        X"0000000", -- 1
                                        X"0000000", -- 2
                                        X"0000000", -- 3
                                        X"0000000", -- 4
                                        X"0000000", -- 5
                                        X"0000000", -- 6
                                        X"0000000", -- 7
```

-- Address $F8 to $FF

```
                                    •
                                    •
                                    •
                                    X"0000000", -- 8
                                    X"0000000", -- 9
                                    X"0000000", -- A
                                    X"0000000", -- B
                                    X"0000000", -- C
                                    X"0000000", -- D
                                    X"0000000", -- E
                                    X"0000000"  -- F
                                    );
    variable addr : integer;
    variable control_out : std_logic_vector(27 downto 0);
```


Begin (process) LSB


```
    begin
        addr := conv_integer(IN_CAR);
        control_out := control_mem(addr);
        MW <= control_out(0);
        MM <= control_out(1);
        RW <= control_out(2);
        MD <= control_out(3);
        FS <= control_out(8 downto 4);
        MB <= control_out(9);
```


Begin (process) MSB


```
        TB <= control_out(10);
        TA <= control_out(11);
        TD <= control_out(12);
        PL <= control_out(13);
        PI <= control_out(14);
        IL <= control_out(15);
        MC <= control_out(16);
        MS <= control_out(19 downto 17);
        NA <= control_out(27 downto 20);
        end process;
    end Behavioral;
```


Control Word forDatapath

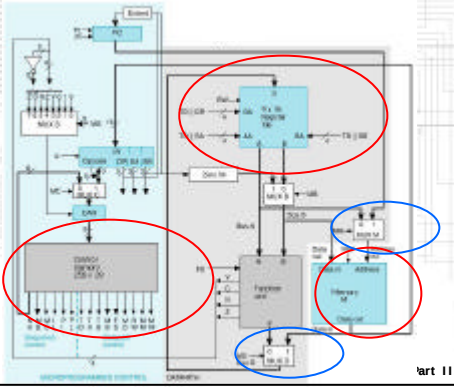
- The Datapath should have the following functionality:

TD	TA	TB	MB		FS		MD	RW	MM	MW	
Select	Select	Select	Select	Code	Function	Code	Select	Function	Select	Function	Code
R[DR]	R[SA]	R[SB]	Register	0	$F = A$	00000	Fn(3)	No write (NW)	Address	No write (NW)	0
IS	IS	IS	Constant	1	$F = A + 1$	00001	Data In	Write (WR)	PC	Write (WR)	1
					$F = A + B$	00010					
					$F = A + \overline{B} + 1$	00011					
					$F = A + \overline{B}$	00100					
					$F = A + B + 1$	00101					
					$F = A - 1$	00110					
					$F = A$	00111					
					$F = A \wedge B$	01000					
					$F = A \vee B$	01001					
					$F = A \oplus B$	01100					
					$F = \overline{A}$	01110					
					$F = B$	10000					
					$F = 0 B$	10100					
					$F = 0 B$	11000					

VHDL top-level models

- The Modified register-file
- The Functional Unit
- The two memories
  - must be implemented as VHDL top-level models and symbols must be created for these components. A schematic should then interconnect these symbols.

Block Diagram



Incomplete Schematic