University of Jordan
Computer Engineering Department
CPE439: Computer Design Lab

Experiment 6: Data Memory Module

It is required to construct and test a Verilog module for a data memory suitable for incorporation in your PIC16F84A design. You should use modular design where you start by building and testing low-level modules using the library modules defined in **Lib439.v**, then use the low-level modules in larger modules. You can also reuse some of the modules that you have designed in previous experiments.

Data Memory

The data memory is partitioned into two areas. The first is the Special Function Registers (SFR) area, while the second is the General Purpose Registers (GPR) area. The SFRs control the operation of the device.

Portions of data memory are banked. The banked areas of the SFR are for the registers that control the peripheral functions. Banking requires the use of control bit (RP0) for bank selection. This control bit is located in the STATUS Register. Figure 1 shows the data memory map organization.

The entire data memory can be accessed either directly using the absolute address of each register file or indirectly through the File Select Register (FSR). Indirect addressing occurs when location 00h is accessed.

The data memory is partitioned into two banks which contain the general purpose registers and the special function registers. Bank 0 is selected by clearing the RP0 bit (STATUS<5>). Setting the RP0 bit selects Bank 1. Each Bank extends up to 4Fh. The first twelve locations of each Bank are reserved for the Special Function Registers. The remainder are General Purpose Registers.

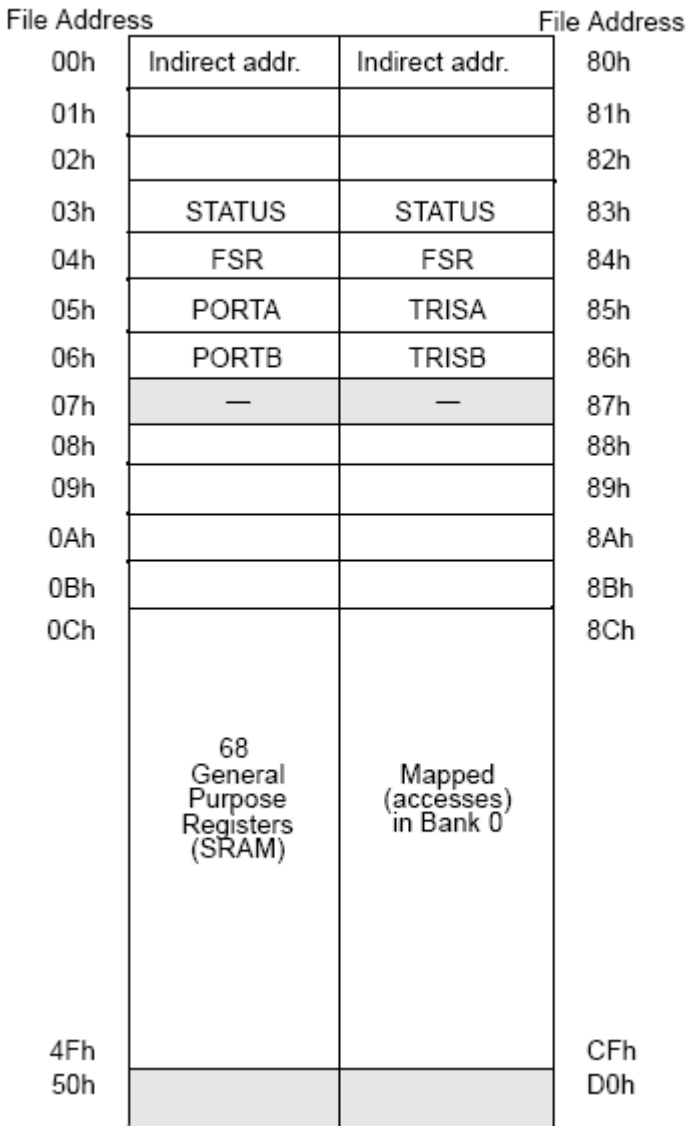


Figure 1: Data Memory Map Organization

Each General Purpose Register (GPR) is 8-bits wide and is accessed either directly or indirectly through the FSR. The GPR addresses in Bank 1 are mapped to addresses in Bank 0. As an example, addressing location 0Ch or 8Ch will access the same GPR. The Special Function Registers (Table 1) are used by the CPU and Peripheral functions to control the device operation.

Table 1: Special Function Registers

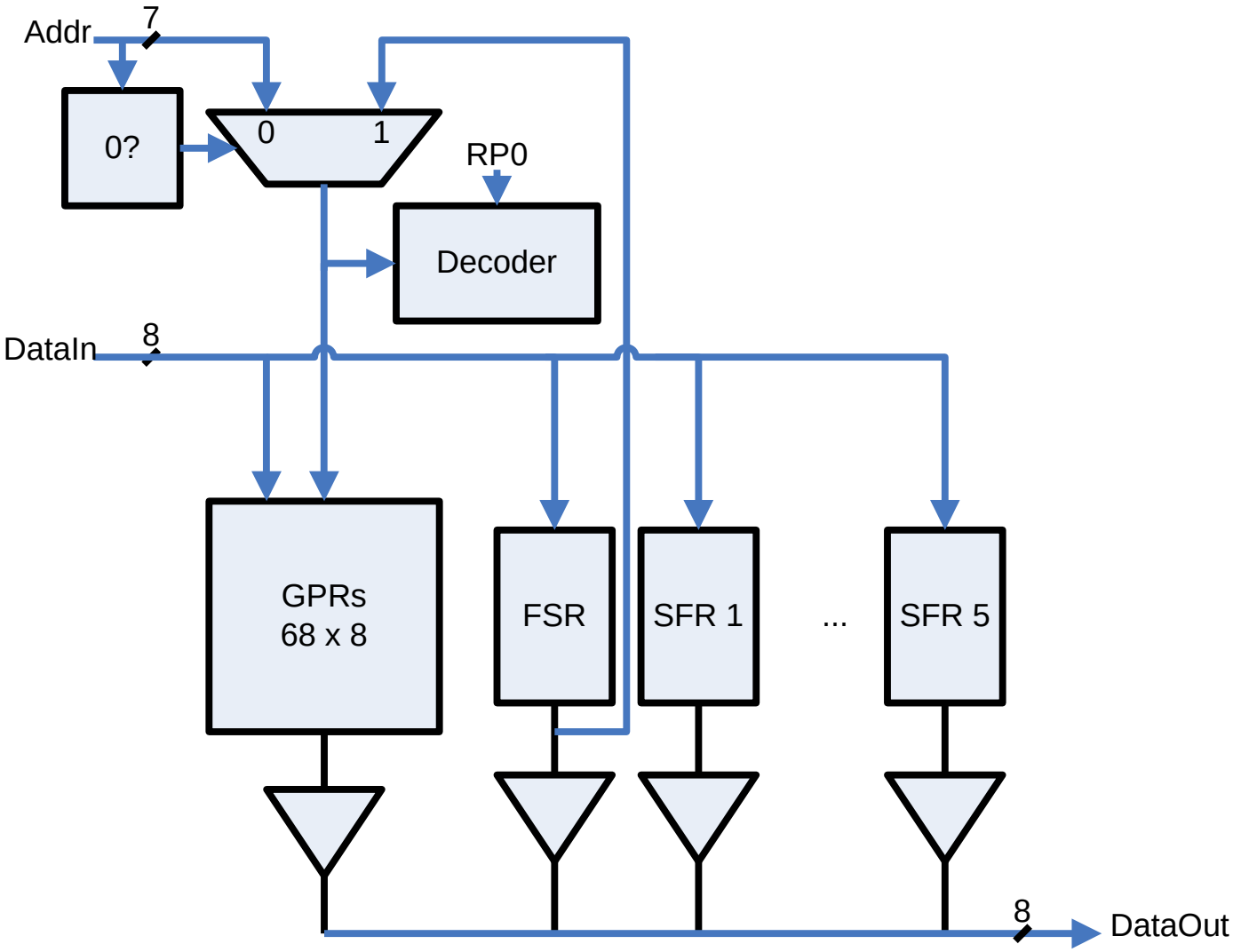
Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on RESET
Bank 0										
00h	INDF	Uses contents of FSR to address Data Memory (not a physical register)								---- ----
01h										
02h										
03h	STATUS(2)			RP0			Z	DC	C	--0- -xxx
04h	FSR	Indirect Data Memory Address Pointer 0								xxxx xxxx
05h	PORTA(4)	—	—	—	RA4	RA3	RA2	RA1	RA0	---x xxxx
06h	PORTB(5)	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx
07h	—	Unimplemented location								—
08h										
09h										
0Ah										
0Bh										
Bank 1										
80h	INDF	Uses Contents of FSR to address Data Memory (not a physical register)								---- ----
81h										
82h										
83h	STATUS (2)			RP0			Z	DC	C	--0- -xxx
84h	FSR	Indirect data memory address pointer 0								xxxx xxxx
85h	TRISA	—	—	—	PORTA Data Direction Register					---1 1111
86h	TRISB	PORTB Data Direction Register								1111 1111
87h	—	Unimplemented location, read as '0'								—
88h										
89h										
0Ah										
0Bh										

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0', q = value depends on condition

This module should have Verilog code similar to the following code:

```
module DataMemory(DataOut, PortA, PortB, C, DC, Z,
                  Clock, Reset, C_in, DC_in, Z_in, C_en, DC_en, Z_en,
                  Addr, DataIn, DataWrite);
  output [7:0] DataOut;
  inout [4:0] PortA;
  inout [7:0] PortB;
  output C, DC, Z;
  input Clock, Reset, C_in, DC_in, Z_in, C_en, DC_en, Z_en;
  input [6:0] Addr;
  input [7:0] DataIn;
  input DataWrite;
  // implementation details are left to the student
  …
endmodule
```

The following figure shows the top-level design of this data memory module.



In this experiment, you need to implement address multiplexer, decoder circuit, GPRs, SFRs, and tri-state butters. The associate circuits for the SFRs should be left to Experiment 7.

General Purpose Registers

The GPRs should be implemented using behavioral modeling as follows:

```
//
// General Purpose Registers, 68 x 8 bits
//
module GPRs(Dout, clock, wt, addr, Din);
   output [7:0] Dout;
   reg [7:0] Dout;
   input clock, wt;
   input [6:0] addr;
   input [7:0] Din;

   reg [7:0] MA [79:0];     //storage array

   always @(addr)
     if ((addr > 7'h0B) && (addr < 7'h50))
       #6 Dout = MA[addr];
```

```
    always @(posedge clock)
      if ((wt == 1) && (addr > 7'h0B) && (addr < 7'h50))
        #1 MA[addr] = Din;

endmodule
```

Report

Your report should include detailed design, Verilog code for all modules including your test modules, and timing diagram that demonstrates the correct operation of your design.