ECE 354 – Computer Systems Lab II

Memory and Indirect Addressing

UMassAmherst

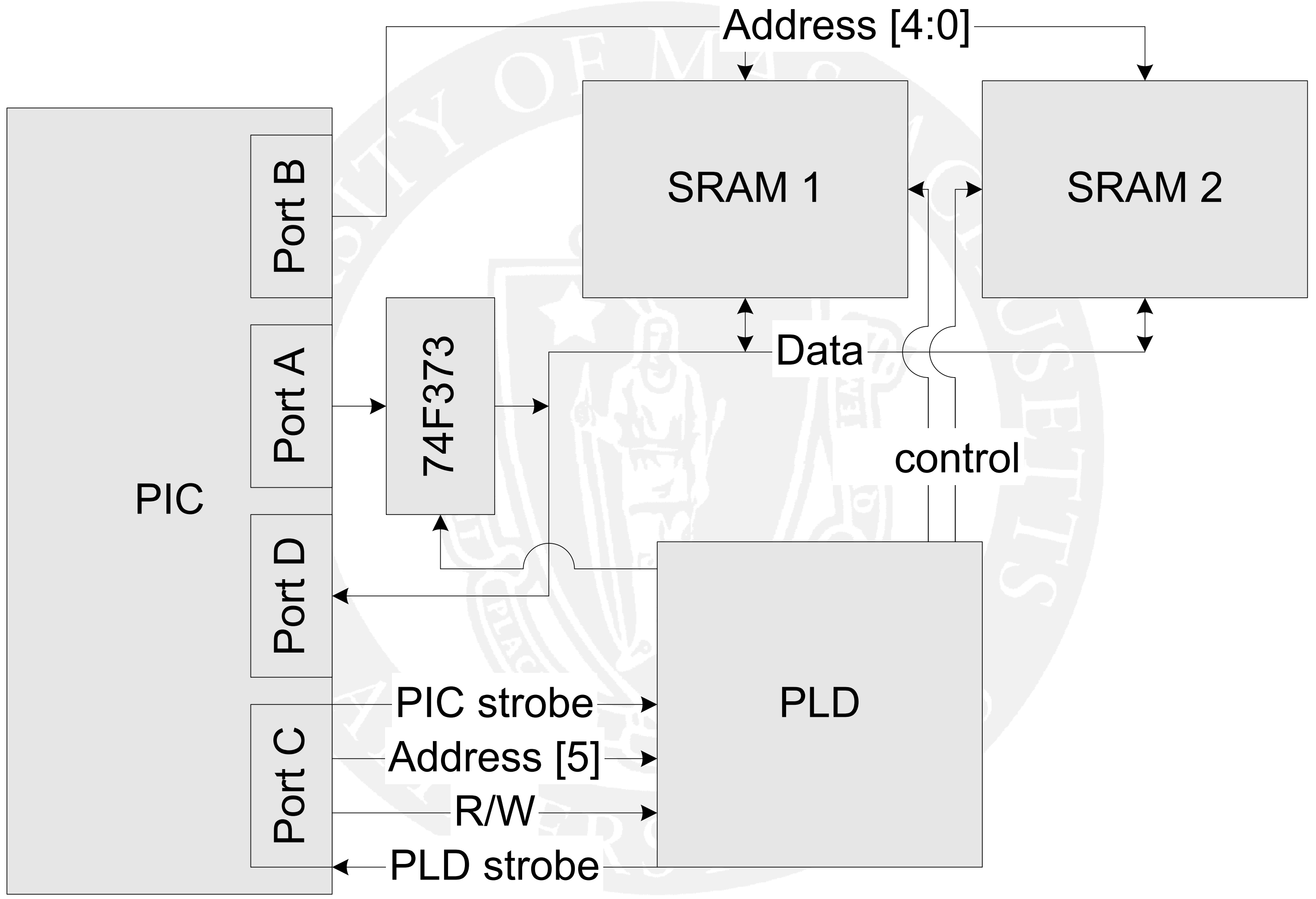
Comments

- Lab report for lab 1
  - Schematics
    - Label pins used on all chips
    - Use standard chip names/numbers (DB25,SP-233) from the datasheet
    - Use standard symbols for LEDs, switches, voltages, ground
    - Label clock frequencies, voltage values (eg. V_{dd}=5V)
    - Straight lines for wires
    - Show orientation of chips
- Lab 2
  - Any problems?
  - Sign up for demo time slot
  - Include logic analyzer outputs in lab report

Lab 3 Overview

- Connect SRAM to PIC
  - SRAM memory extends storage of system
- Set up virtual address space
  - Spans PIC and two SRAM chips
- Memory test routine
  - Checks if write/read to memory works
  - Test result shown on terminal
- User can specify:
  - Address range
  - Test pattern to be used
- Each test reads and writes eight test patterns (rotated each time) to memory

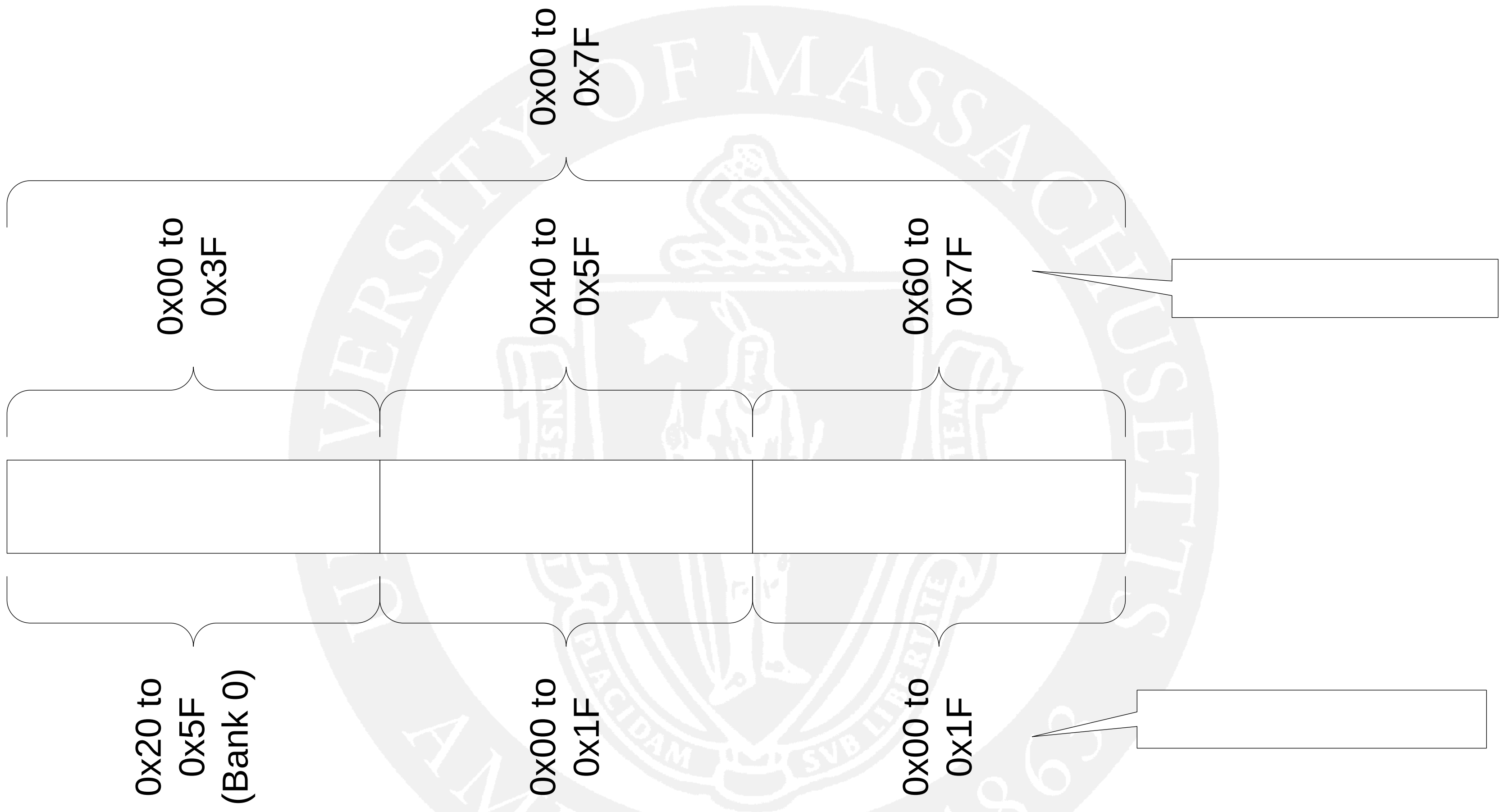
Lab 3 System Outline



Address Mapping

- Address space of Lab 3 spans three chips
  - Systems might require multiple memory chips (why?)
- Address space should be simple
  - Simpler to program
  - Independent of physical memory configuration
- "Memory Mapping"
  - Assignment of logical/virtual addresses to physical memory addresses
- Lab 3:
  - Three memories: 2 SRAM chips, internal PIC data memory
  - Address space: 128 bytes 0x00 to 0x7F

Address Map Example

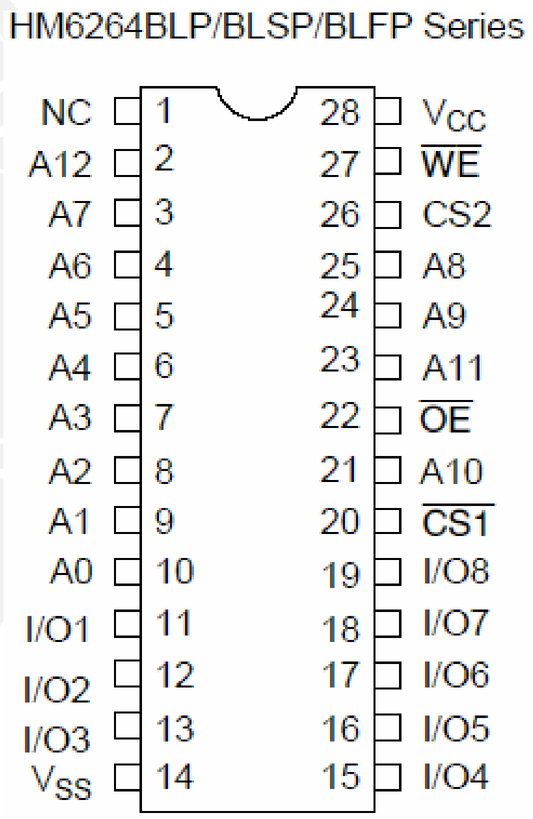


Address Decoding

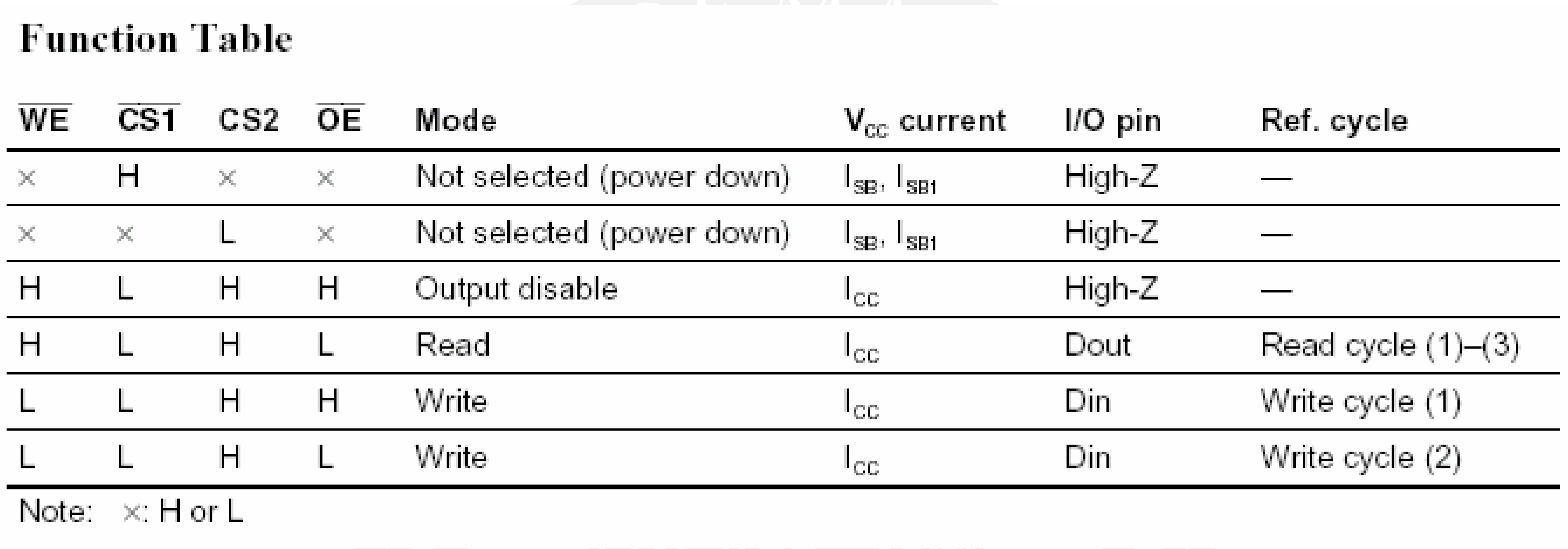
- Address decoding is process of determining to which device address is mapped
- Full address decoding
  - Look at every bit in the address
  - Requires more hardware
  - Possibly slower decoding
  - Can perform out-of-range checking
- Partial address decoding
  - Uses only some of the address bits to select device
  - Use some bits to select address within device
  - Cannot perform arbitrary range checking
  - Needs careful design
  - Uses fewer pins/gates
- What are we using in Lab 3?

SRAM

- "Static Random Access Memory"
  - Memory does not need to be refreshed (unlike SDRAM)
- We are using HM6264B for Lab 3
  - 64kbit SRAM (8k x 8 bit)
  - Effectively only very small fraction of SRAM is used
- Important pins:
  - A0 to A12: address
  - I/O1 to I/O8: data input/output
  - CS1 and CS2: chip select 1 and 2
  - WE: write enable
  - OE: output enable



SRAM Functions



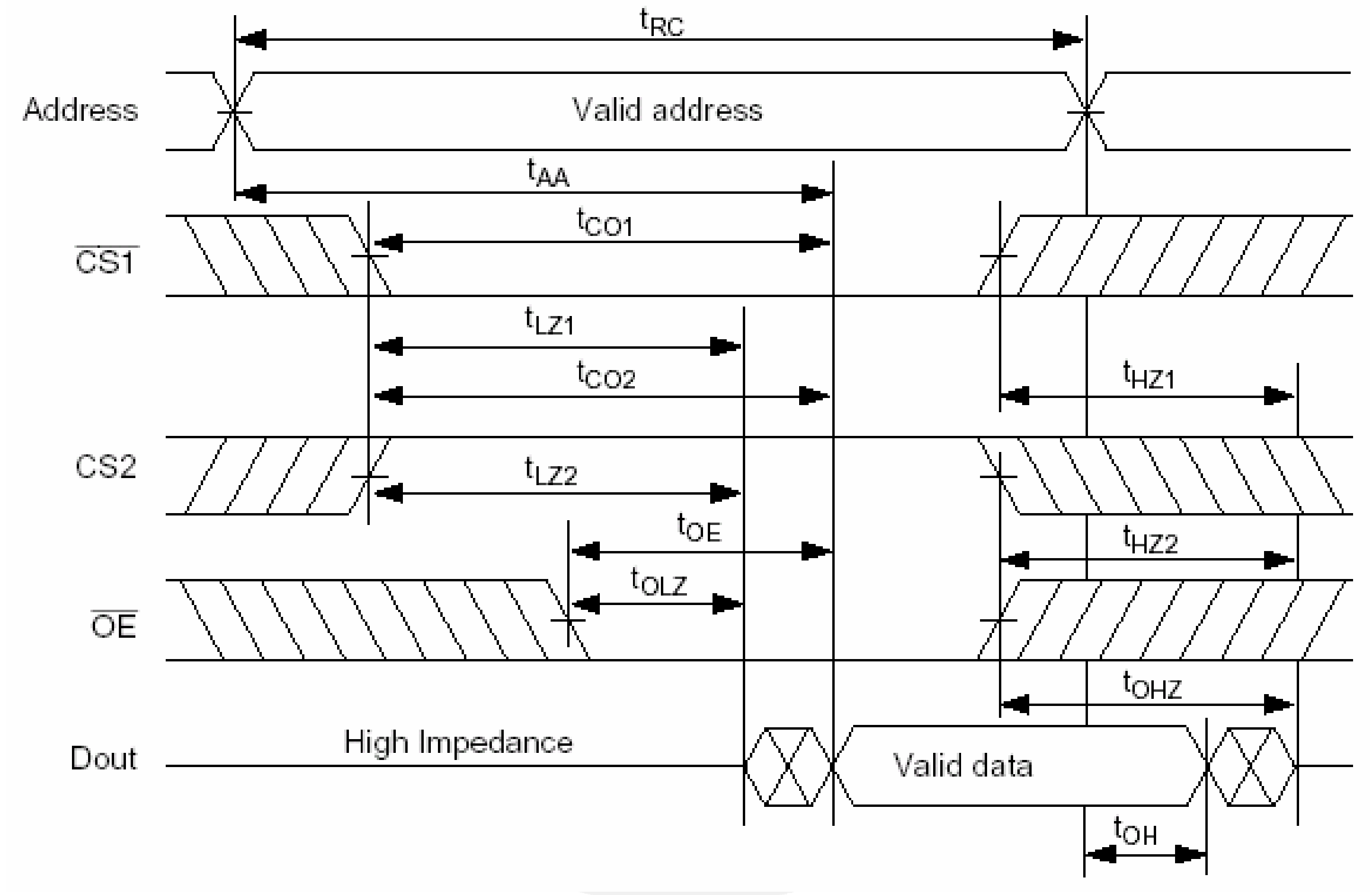
Function Table

$\overline{WE}$	$\overline{CS1}$	CS2	$\overline{OE}$	Mode	V_{CC} current	I/O pin	Ref. cycle
×	H	×	×	Not selected (power down)	I_{SB}, I_{SB1}	High-Z	—
×	×	L	×	Not selected (power down)	I_{SB}, I_{SB1}	High-Z	—
H	L	H	H	Output disable	I_{CC}	High-Z	—
H	L	H	L	Read	I_{CC}	Dout	Read cycle (1)–(3)
L	L	H	H	Write	I_{CC}	Din	Write cycle (1)
L	L	H	L	Write	I_{CC}	Din	Write cycle (2)

Note: ×: H or L

SRAM Read (1)

- Assuming R/W* is asserted:

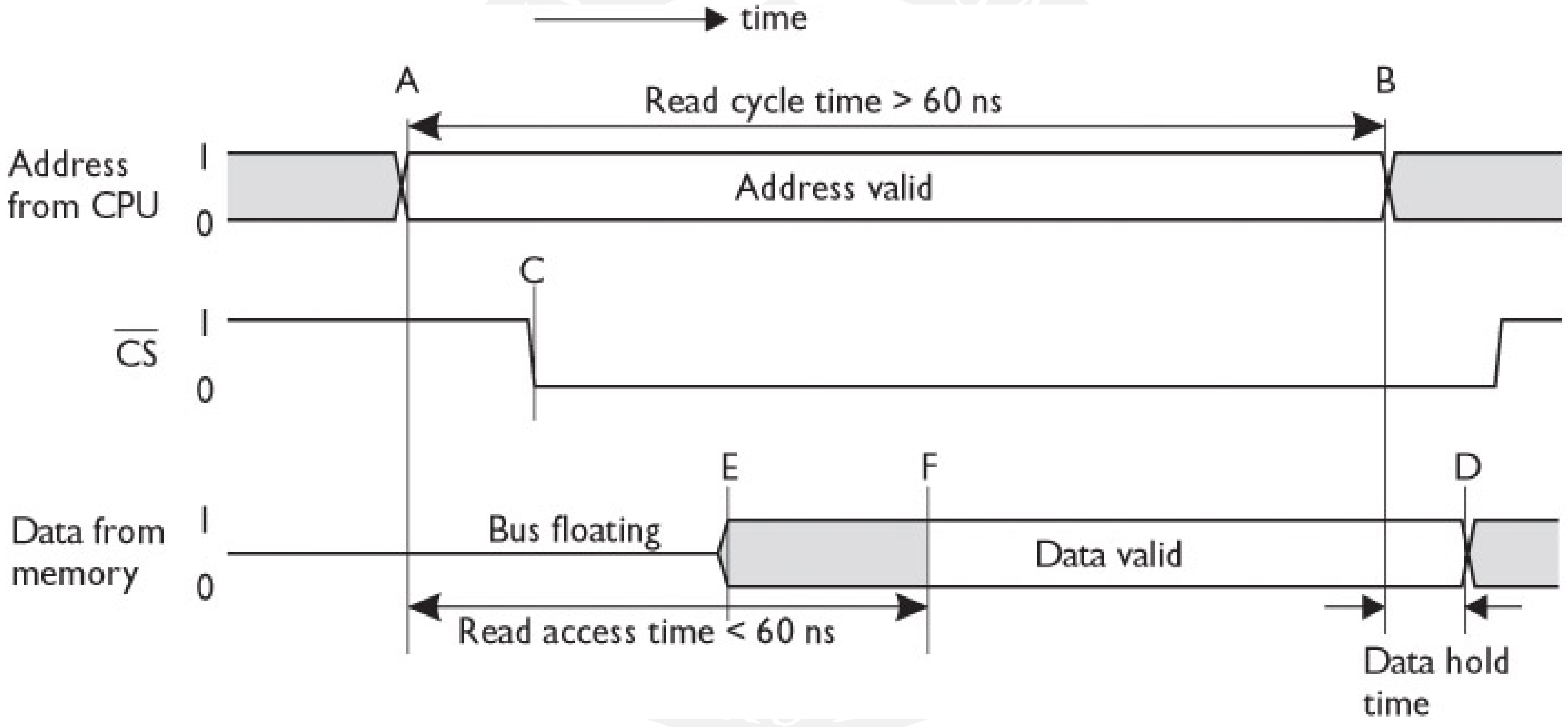


SRAM Read (2)

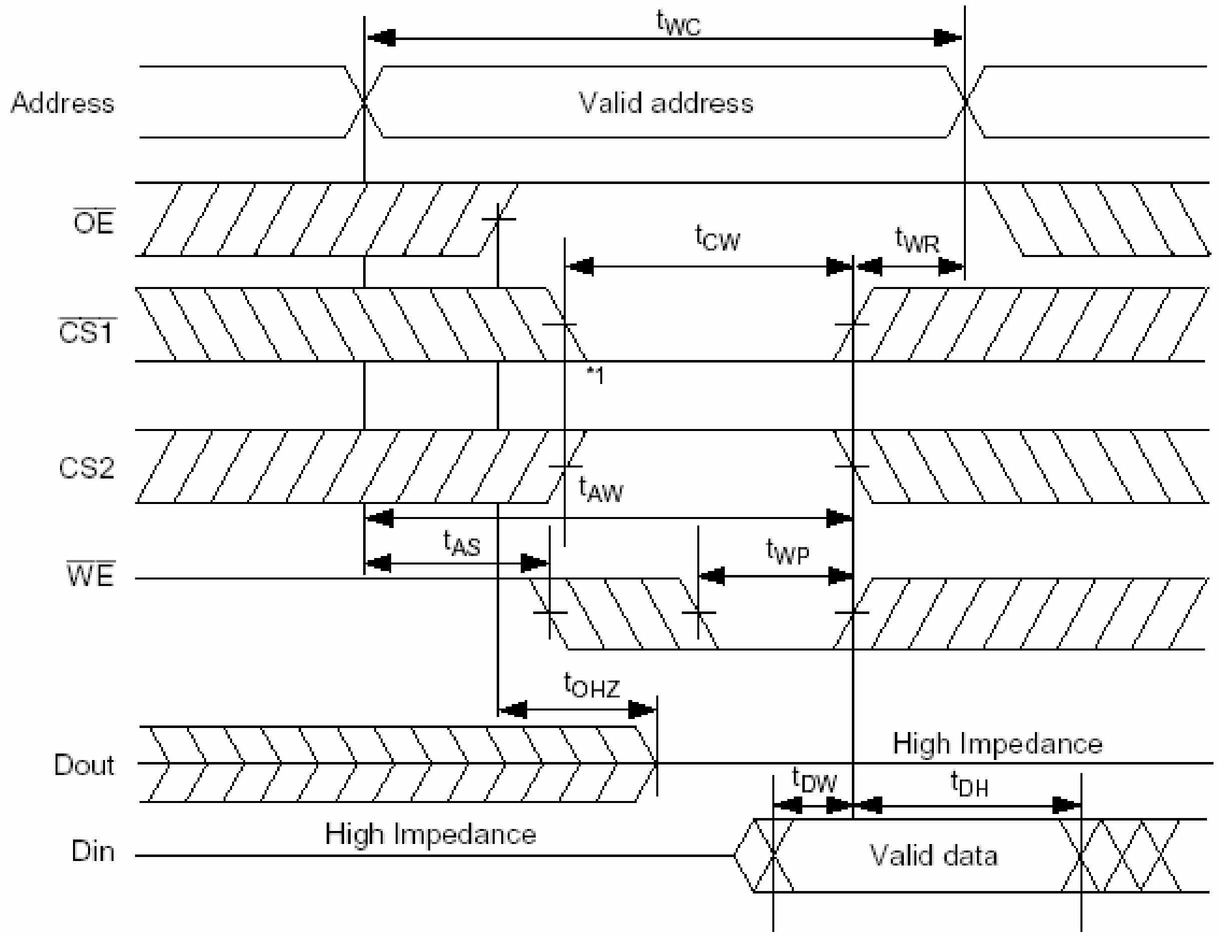
1. WE* asserted during transaction
2. Address asserted
3. CS*, OE* deasserted
4. Data available (transfer to PIC)
5. CS*, OE* asserted
6. Address deasserted

SRAM Read (3)

- From Alan Clements: "The Principles of Computer Hardware," Third Edition, Oxford University Press



SRAM Write (1)

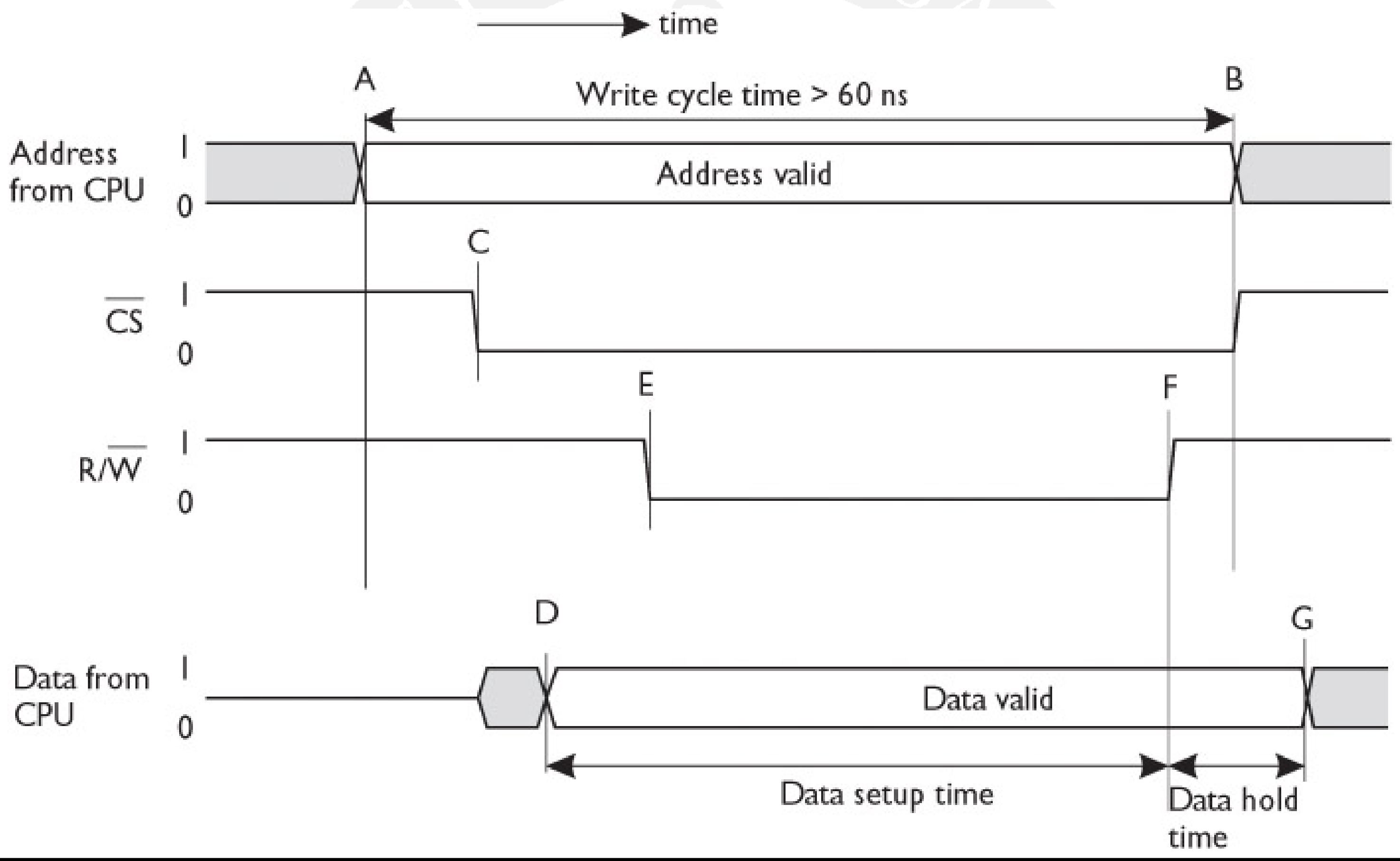

t_WC
Address
Valid address
OE
t_CW
t_WR
CS1
*1
CS2
t_AW
t_AS
t_WP
WE
t_OHZ
Dout
High Impedance
t_DW
t_DH
Din
High Impedance
Valid data

SRAM Write (2)

1. OE* asserted during transaction
2. Address asserted
3. CS* deasserted
4. WE* deasserted
5. Data driven by PIC (anytime during 1-4)
6. CS*, WE* asserted
7. Data and address deasserted

SRAM Write (3)

- From Alan Clements: “The Principles of Computer Hardware,” Third Edition, Oxford University Press

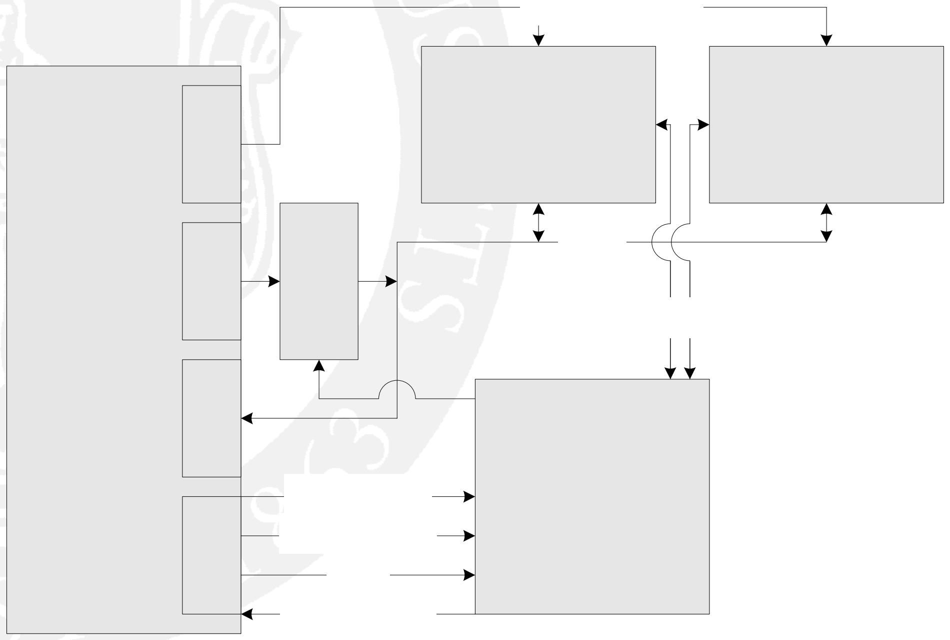


SRAM Hints

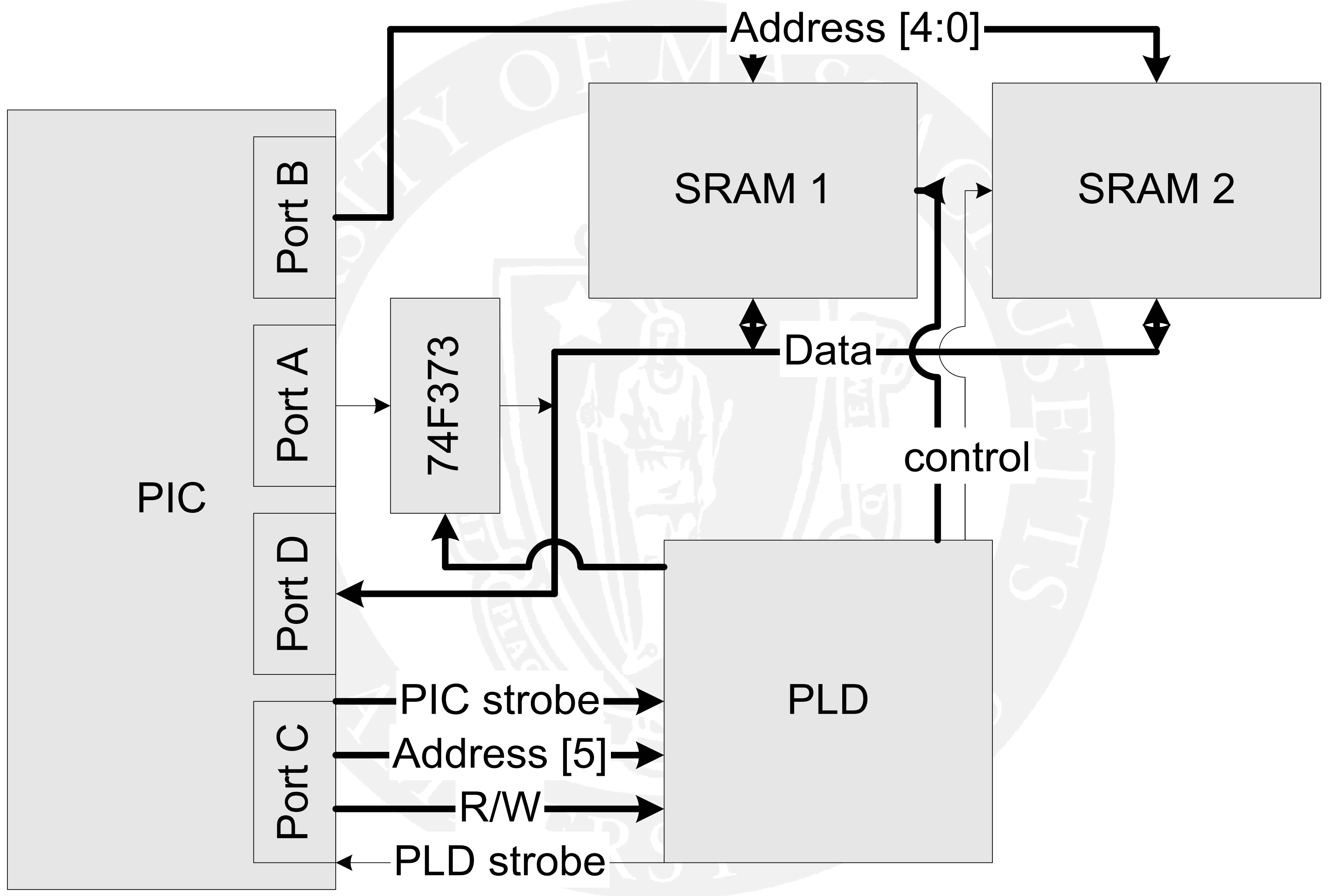
- No harm in asserting address/data early
- Unused address pins
  - Drive to ground
- Unused data pins (only during test)
  - Tie to ground through 4.7k resistor
- With multiple SRAMs on one bus
  - only one OE should be enabled at any time
- I/O pins are bidirectional
  - Separate to two PIC ports
  - Isolate PIC output port with tri-state buffer (74F373 chip)
- SRAM is self-timed
- Do not confuse inverted control signals
- For details on memory interface, read Chapter 9 in Alan Clements: “The Principles of Computer Hardware,” Third Edition, Oxford University Press (available online, link from course page)

Putting It All Together

- SRAM is controlled by PLD
  - PLD implements control of OE*, WE*, CS*
- PIC uses bus (Lab 2) to communicate with PLD
  - Data and address go directly from PIC to SRAM and back
- Address is split between PLD and SRAM

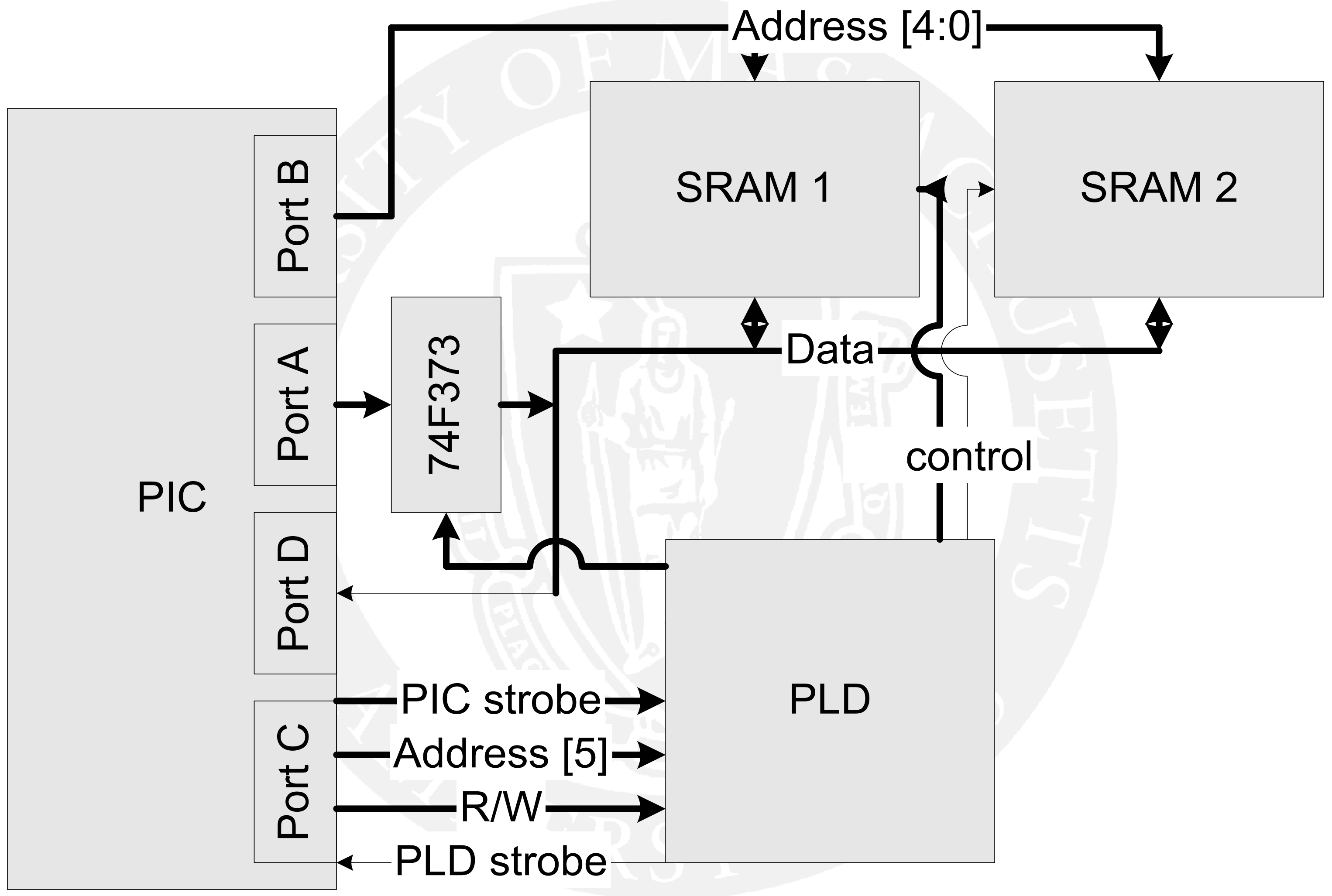


Lab 3 Read



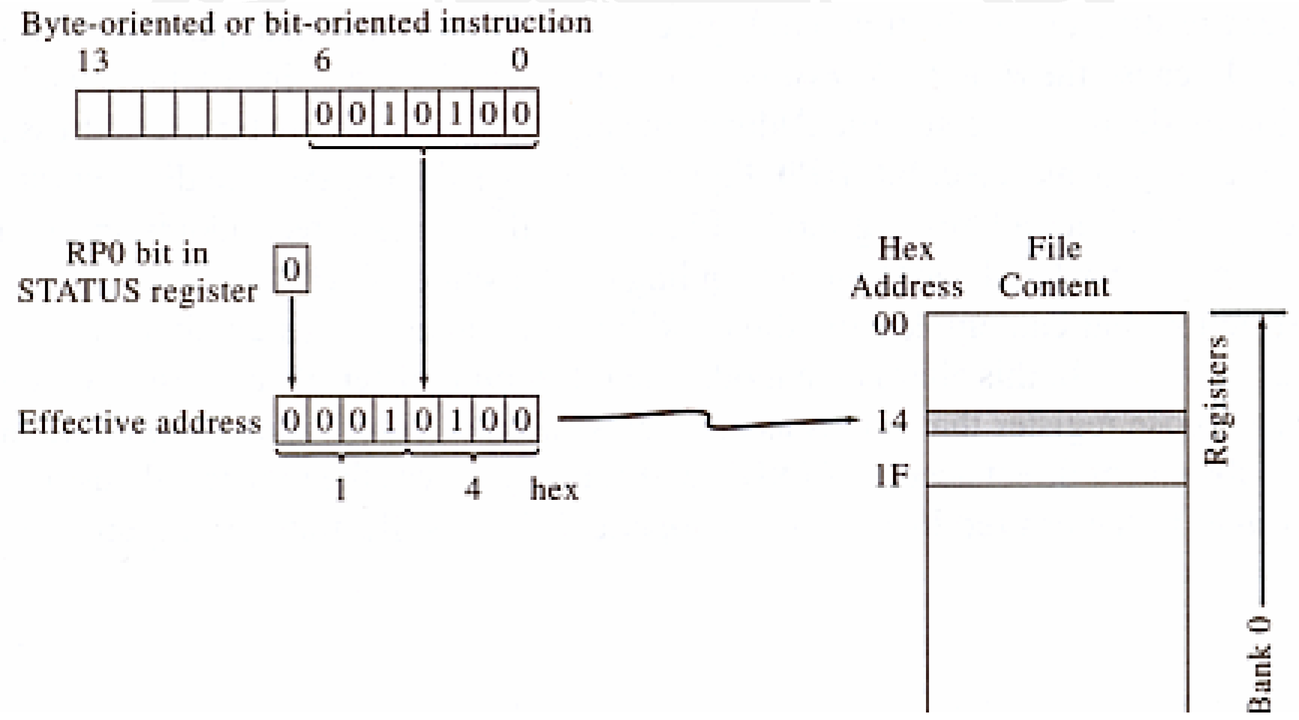
Address [4:0]
SRAM 1
SRAM 2
Port B
Port A
74F373
Data
control
PIC
Port D
PLD
Port C
PIC strobe
Address [5]
R/W
PLD strobe

Lab 3 Write


Address [4:0]
SRAM 1
SRAM 2
Port B
Port A
74F373
Data
control
PIC
Port D
PLD
Port C
PIC strobe
Address [5]
R/W
PLD strobe

Internal Memory Test

- Memory test needs to access range of addresses
- Problem: in programs addresses are fixed
  - Part of the instruction (“direct addressing”)
  - Programmed into instruction memory



Accessing Memory Range

- Example: write 0x42 to registers 0x20 to 0x7f

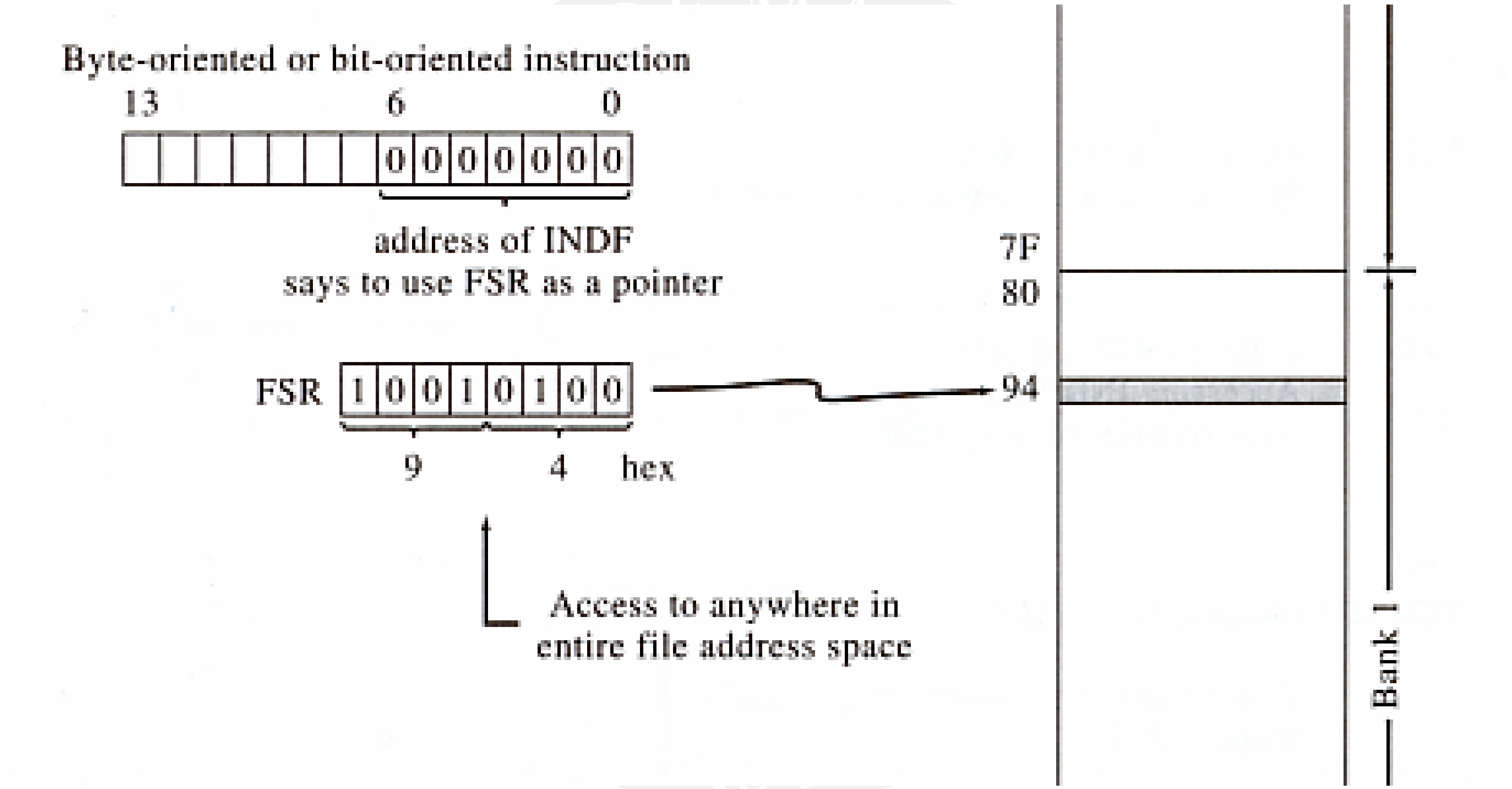
```
TEMP20   EQU    0x20
TEMP21   EQU    0x21
...
TEMP7F   EQU    0x7F


main     movlw  0x42
         movwf  TEMP20
         movwf  TEMP21
...
         movwf  TEMP7F
```

- Clumsy, wasteful – need better method!

Indirect Addressing (1)

- Indirect Addressing:
  - Do not fix address in instruction
  - Use address given in another register
- PIC uses INDF and FSR registers
  - FSR contains address of register (8 bits)
  - IRP bit (STATUS<7>) selects bank 0/1 or bank 2/3 (1 bit)
  - Read or write to INDF register will automatically perform indirect access

Indirect Addressing (2)


Byte-oriented or bit-oriented instruction
13
6
0
0 0 0 0 0 0 0
address of INDF
says to use FSR as a pointer
FSR
1 0 0 1 0 1 0 0
9
4
hex
Access to anywhere in
entire file address space
7F
80
94
Bank 1

Accessing Memory Range

- Example: write 0x42 to registers 0x20 to 0x7f

```
      movlw 0x20   ; initialize first address
      movwf FSR    ; set indirection register
main  movlw 0x42   ; move test value to W register
      movwf INDF   ; move value to indirect reg
      incf  FSR,F ; point to next data location
....               ; test if 0x80 reached
      btfss STATUS, Z ; if yes, then leave loop
      goto  main   ; restart loop
....               ; rest of code here
```

Final Comments

- Start early – really!
  - Demos are on April 15th and 16th
  - Rob Vice's lab hours will end after this week
- Use logic analyzer from the start
- Lab is difficult, because many components involved
  - Structure your breadboard layout
- Spend some time on your design
  - Don't reuse Lab 2 bus blindly
- It's the most difficult project – it'll get easier after this