

---

Computer Organization & Design  
Chapter Five  
Sec. 5.5 - 5.11  
Microcode for Control

1

---

---

---

---

---

---

---

---

---

The Processor: Datapath & Control

- We're ready to look at an implementation of the MIPS
- Simplified to contain only:
  - memory-reference instructions: `lw, sw`
  - arithmetic-logical instructions: `add, sub, and, or, slt`
  - control flow instructions: `beq, j`
- Generic Implementation:
  - use the program counter (PC) to supply instruction address
  - get the instruction from memory
  - read registers
  - use the instruction to decide exactly what to do
- All instructions use the ALU after reading the registers  
Why? memory-reference? arithmetic? control flow?

2

---

---

---

---

---

---

---

---

---

Control

- e.g., what should the ALU do with this instruction
- Example: `lw $1, 100($2)`

35	2	1	100
----	---	---	-----

op	rs	rt	16 bit offset
----	----	----	---------------

- ALU control input

000	AND
001	OR
010	add
110	subtract
111	set-on-less-than
- Why is the code for subtract 110 and not 011?

3

---

---

---

---

---

---

---

---

## Control

- Must describe hardware to compute 3-bit ALU control input
  - given instruction type
    - 00 = lw, sw
    - 01 = beq, 11 = arithmetic
  - function code for arithmetic
- Describe it using a truth table (can turn into gates):

ALUOp		Func field						Operation
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	
0	0	X	X	X	X	X	X	010
X	1	X	X	X	X	X	X	110
1	X	X	X	0	0	0	0	010
1	X	X	X	0	0	1	0	110
1	X	X	X	0	1	0	0	000
1	X	X	X	0	1	0	1	001
1	X	X	X	1	0	1	0	111

4

---

---

---

---

---

---

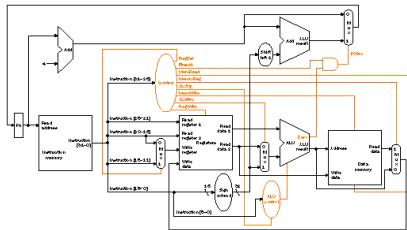
---

---

---

---

## Control



Instruction	RegDst	ALUSrc	Memto-Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp1	ALUOp0
R-format	1	0	0	1	0	0	0	1	0
lwr	0	1	1	1	1	0	0	0	0
lwr	X	1	X	0	0	1	1	0	0
lwr	X	0	X	0	0	0	1	0	1

5

---

---

---

---

---

---

---

---

---

---

## Implementing the Control

- Value of control signals is dependent upon:
  - what instruction is being executed
  - which step is being performed
- Use the information we've accumulated to specify a finite state machine
  - specify the finite state machine graphically, or
  - use microprogramming
- Implementation can be derived from specification

6

---

---

---

---

---

---

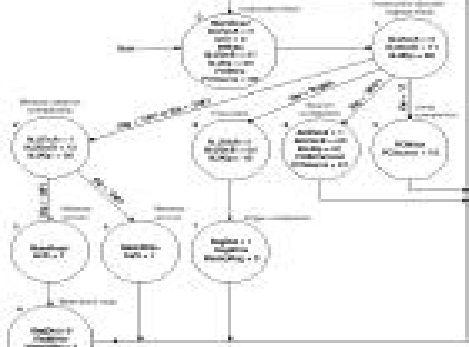
---

---

---

---

### Graphical Specification of FSM



- How many state bits will we need? 10 states,  $< 2^4$  4 bits 7

---

---

---

---

---

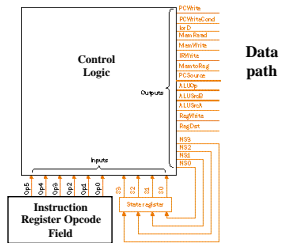
---

---

---

### Finite State Machine for Control

- Implementation:



8

---

---

---

---

---

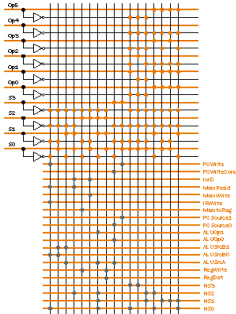
---

---

---

### PLA Implementation

- If I picked a horizontal or vertical line could you explain it?



9

---

---

---

---

---

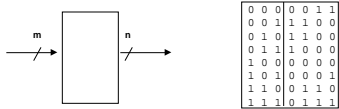
---

---

---

## ROM Implementation

- ROM = "Read Only Memory"
  - values of memory locations are fixed ahead of time
- A ROM can be used to implement a truth table
  - if the address is m-bits, we can address  $2^m$  entries in the ROM.
  - our outputs are the bits of data that the address points to.



m is the "height", and n is the "width"

10

---

---

---

---

---

---

---

---

---

---

## ROM Implementation

- How many inputs are there?  
6 bits for opcode, 4 bits for state = 10 address lines  
(i.e.,  $2^{10} = 1024$  different addresses)
- How many outputs are there?  
16 datapath-control outputs, 4 state bits = 20 outputs
- ROM is  $2^{10} \times 20 = 20K$  bits (and a rather unusual size)
- Rather wasteful, since for lots of the entries, the outputs are the same
  - i.e., opcode is often ignored

11

---

---

---

---

---

---

---

---

---

---

## ROM vs PLA

- Break up the table into two parts
  - 4 state bits tell you the 16 outputs,  $2^4 \times 16$  bits of ROM
  - 10 bits tell you the 4 next state bits,  $2^{10} \times 4$  bits of ROM
  - Total: 4.3K bits of ROM
- PLA is much smaller
  - can share product terms
  - only need entries that produce an active output
  - can take into account don't cares
- Size is (#inputs  $\times$  #product-terms) + (#outputs  $\times$  #product-terms)  
For this example =  $(10 \times 17) + (20 \times 17) = 460$  PLA cells
- PLA cells usually about the size of a ROM cell (slightly bigger)

12

---

---

---

---

---

---

---

---

---

---





### Microcode: Trade-offs

- Distinction between specification and implementation is sometimes blurred
- Specification Advantages:
  - Easy to design and write
  - Design architecture and microcode in parallel
- Implementation (off-chip ROM) Advantages
  - Easy to change since values are in memory
  - Can emulate other architectures
  - Can make use of internal registers
- Implementation Disadvantages, SLOWER now that:
  - Control is implemented on same chip as processor
  - ROM is no longer faster than RAM
  - No need to go back and make changes

19

---

---

---

---

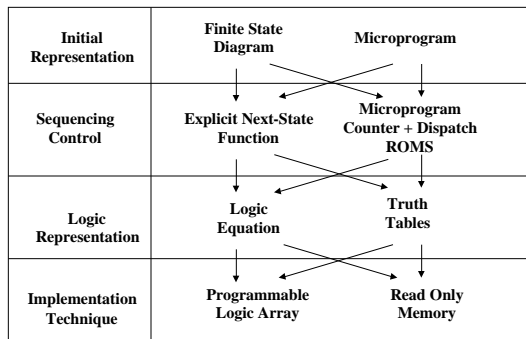
---

---

---

---

### The Big Picture



20

---

---

---

---

---

---

---

---