ECE 361
Computer Architecture
Lecture 11: Designing a Multiple Cycle Controller

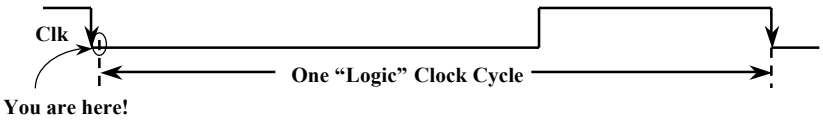
Review of a Multiple Cycle Implementation

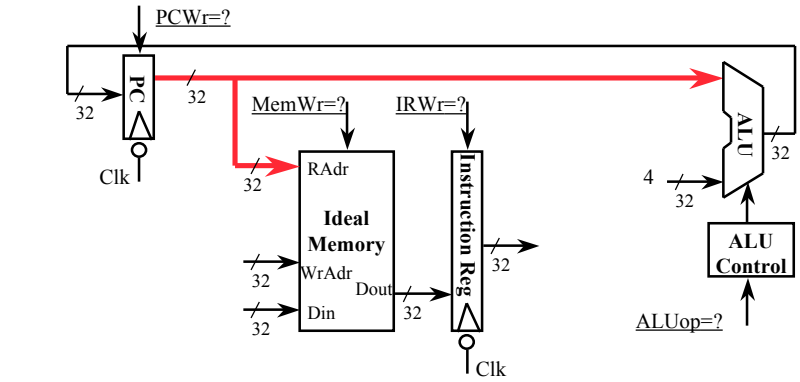
- The root of the single cycle processor's problems:
  - The cycle time has to be long enough for the slowest instruction
- Solution:
  - Break the instruction into smaller steps
  - Execute each step (instead of the entire instruction) in one cycle
    - Cycle time: time it takes to execute the longest step
    - Keep all the steps to have similar length
  - This is the essence of the multiple cycle processor
- The advantages of the multiple cycle processor:
  - Cycle time is much shorter
  - Different instructions take different number of cycles to complete
    - Load takes five cycles
    - Jump only takes three cycles
  - Allows a functional unit to be used more than once per instruction

Review: Instruction Fetch Cycle, In the Beginning

° Every cycle begins right AFTER the clock tick:

- mem[PC] PC<31:0> + 4



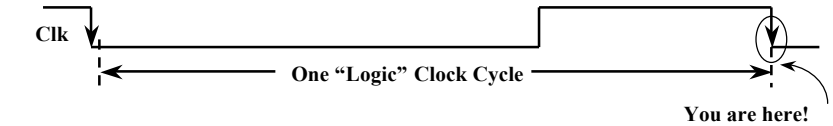


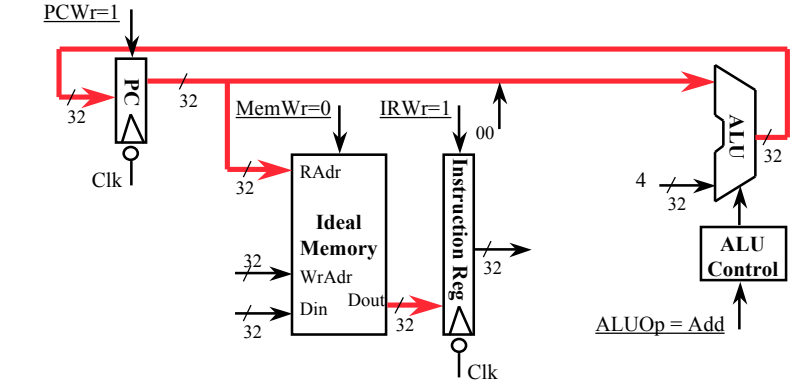
361 multicontroller.3

Review: Instruction Fetch Cycle, The End

° Every cycle ends AT the next clock tick (storage element updates):

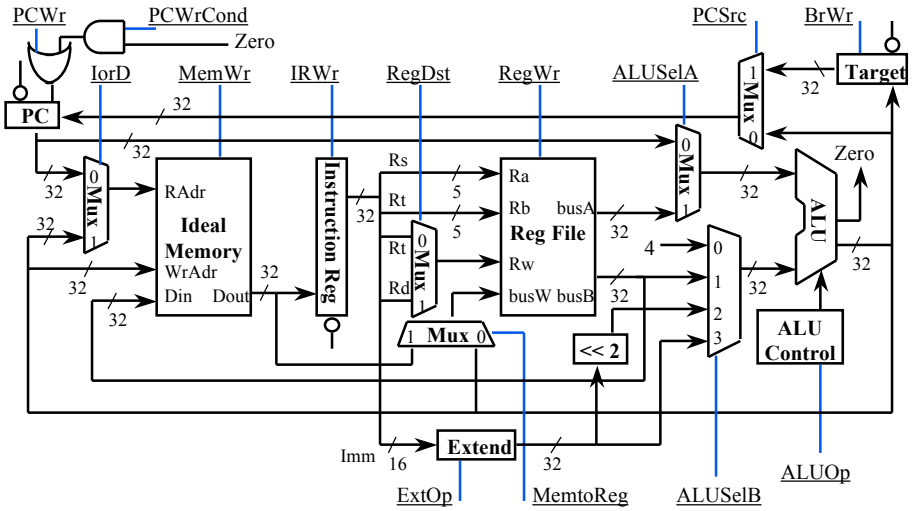
- IR <-- mem[PC] PC<31:0> <-- PC<31:0> + 4



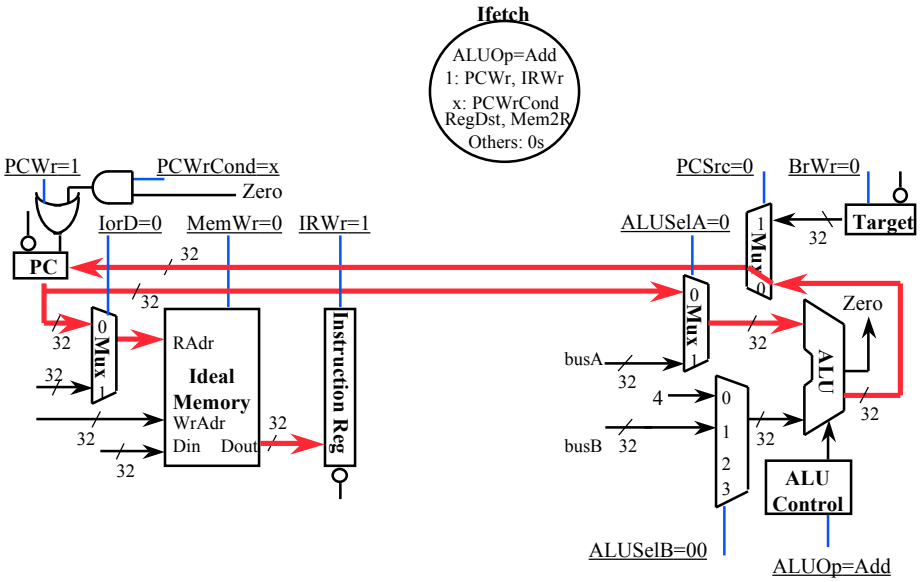


361 multicontroller.4

Putting it all together: Multiple Cycle Datapath



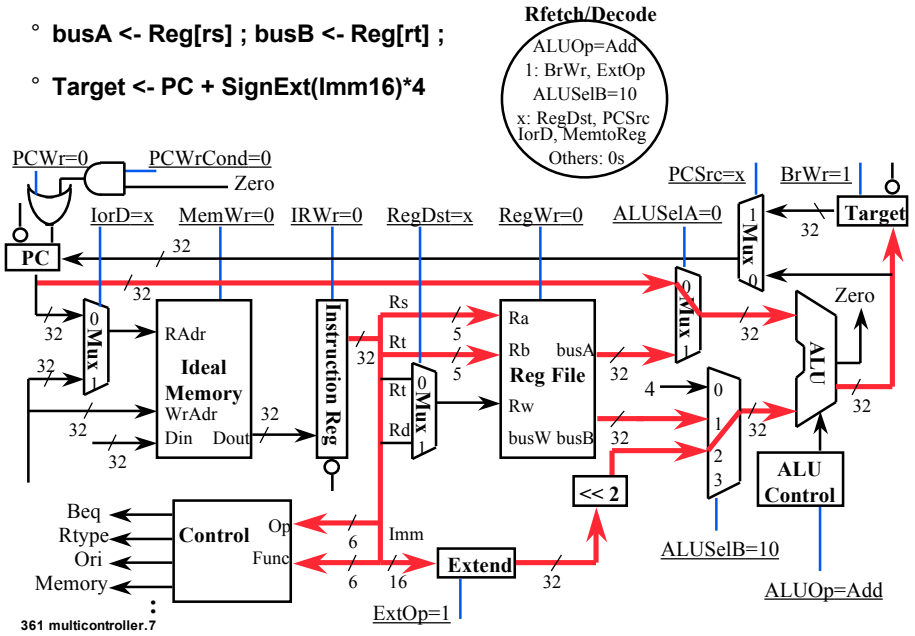
Instruction Fetch Cycle: Overall Picture



Register Fetch / Instruction Decode (Continue)

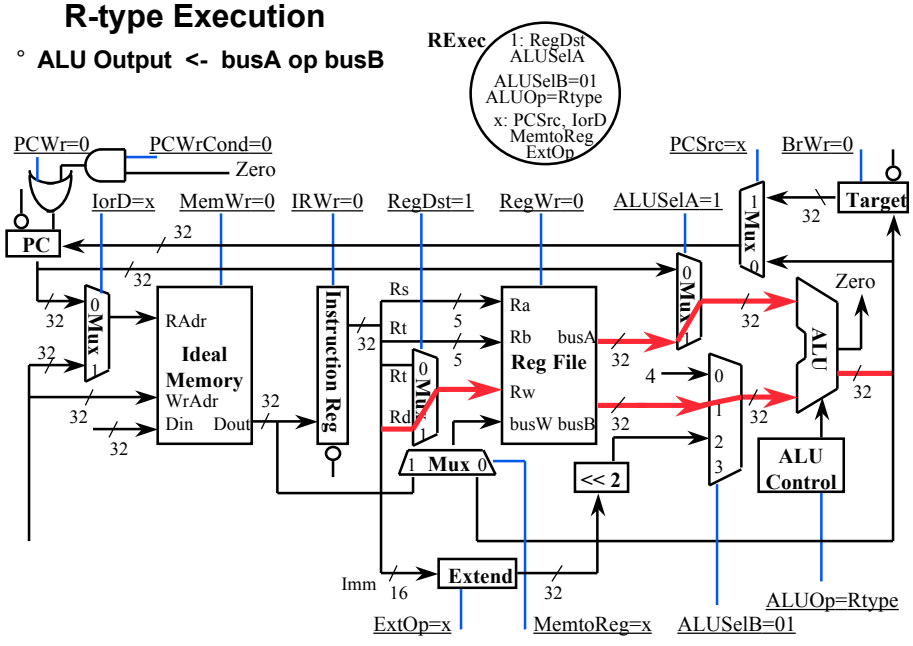
° busA <- Reg[rs] ; busB <- Reg[rt] ;

° Target <- PC + SignExt(Imm16)*4



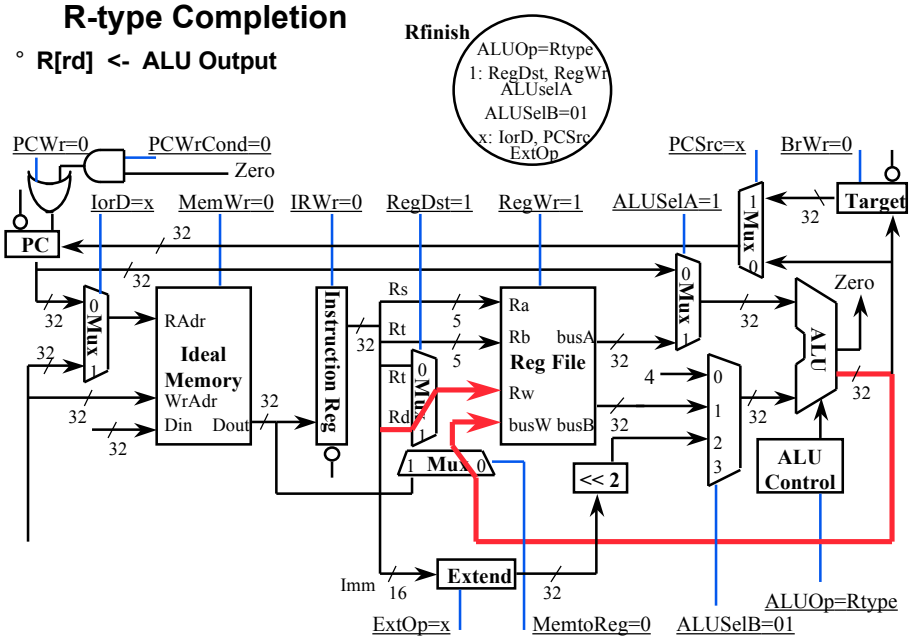
R-type Execution

° ALU Output <- busA op busB



R-type Completion

° R[rd] <- ALU Output



Outline of Today's Lecture

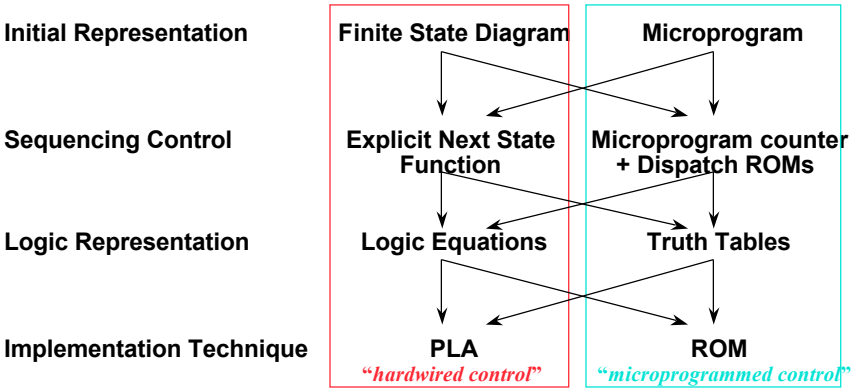
° Recap

° Review of FSM control

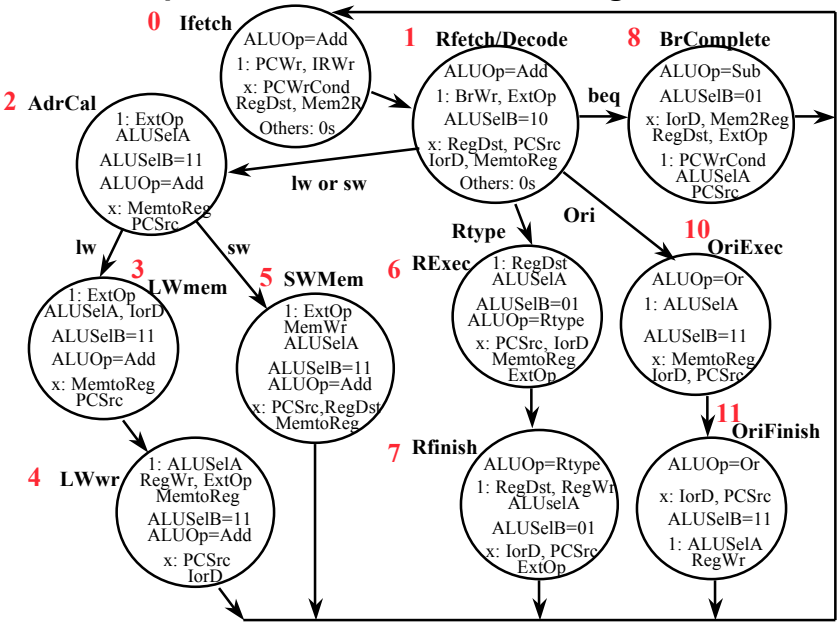
° From Finite State Diagrams to Microprogramming

Overview

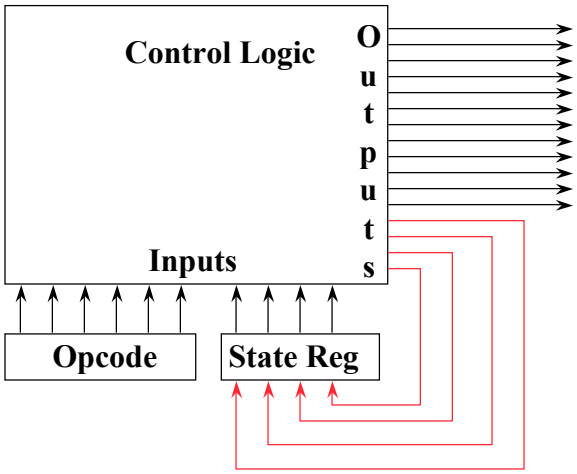
° Control may be designed using one of several initial representations. The choice of sequence control, and how logic is represented, can then be determined independently; the control can then be implemented with one of several methods using a structured logic technique.



Initial Representation: Finite State Diagram



Sequencing Control: Explicit Next State Function



Multicycle Datapath

° Next state number is encoded just like datapath controls

Logic Representative: Logic Equations

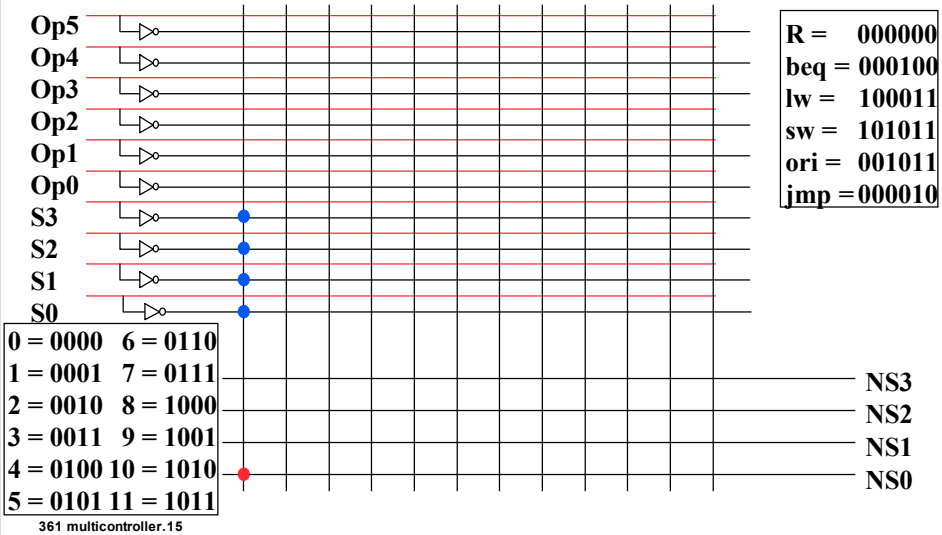
° Next state from current state
- State 0 -> State1
- State 1 -> S2, S6, S8, S10
- State 2 ->_____
- State 3 ->_____
- State 4 ->State 0
- State 5 -> State 0
- State 6 -> State 7
- State 7 -> State 0
- State 8 -> State 0
- State 9-> State 0
- State 10 -> State 11
- State 11 -> State 0

°Alternatively,
prior state & condition

S4, S5, S7, S8, S9, S11 -> State0
_____ -> State 1
_____ -> State 2
_____ -> State 3
_____ -> State 4
State2 & op = sw -> State 5
_____ -> State 6
State 6 -> State 7
_____ -> State 8
State2 & op = jmp -> State 9
_____ -> State 10
State 10 -> State 11

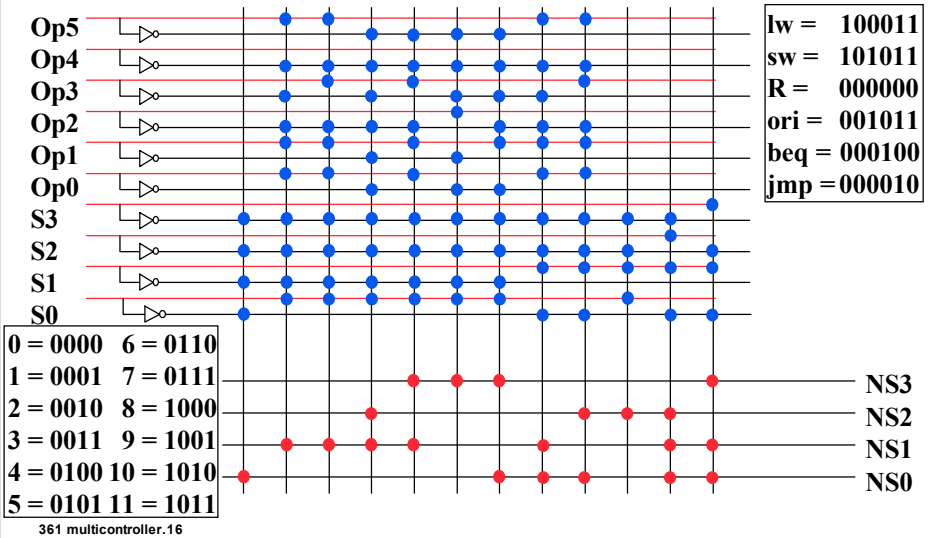
Implementation Technique: Programmed Logic Arrays

° Each output line the logical OR of logical AND of input lines or their complement: AND minterms specified in top AND plane, OR sums specified in bottom OR plane



Implementation Technique: Programmed Logic Arrays

° Each output line the logical OR of logical AND of input lines or their complement: AND minterms specified in top AND plane, OR sums specified in bottom OR plane

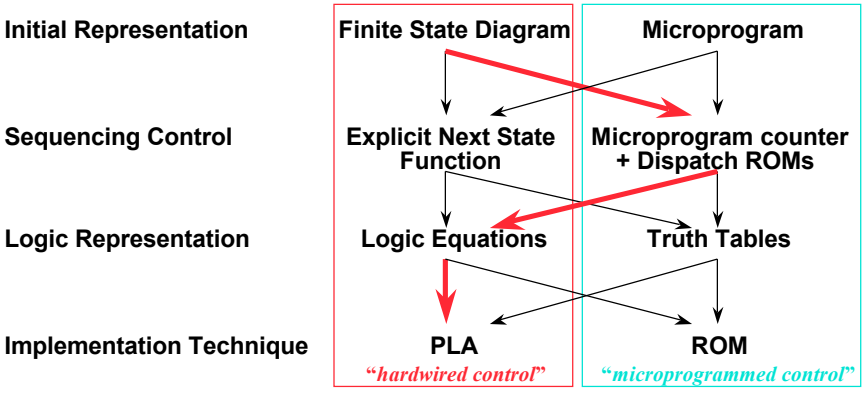


Multicycle Control

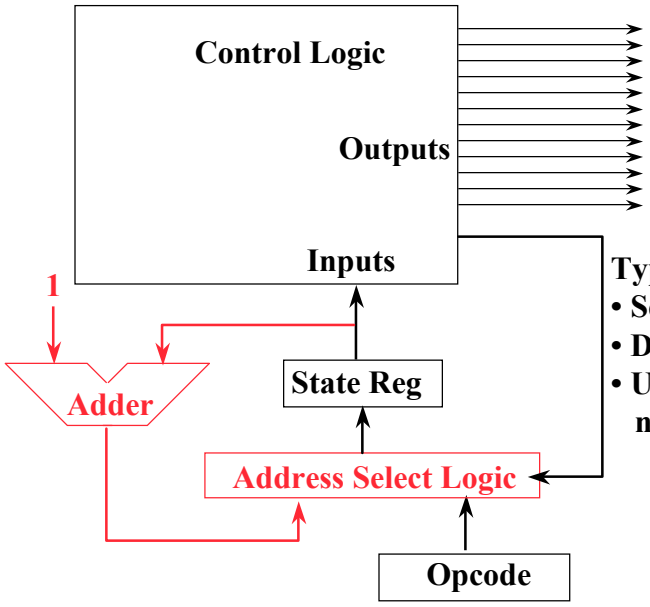
- Given numbers of FSM, can turn determine next state as function of inputs, including current state
- Turn these into Boolean equations for each bit of the next state lines
- Can implement easily using PLA
- What if many more states, many more conditions?
- What if need to add a state?

Next Iteration: Using Sequencer for Next State

- Before Explicit Next State: Next try variation 1 step from right hand side
- Few sequential states in small FSM: suppose added floating point?
- Still need to go to non-sequential states: e.g., state 1 => 2, 6, 8, 10

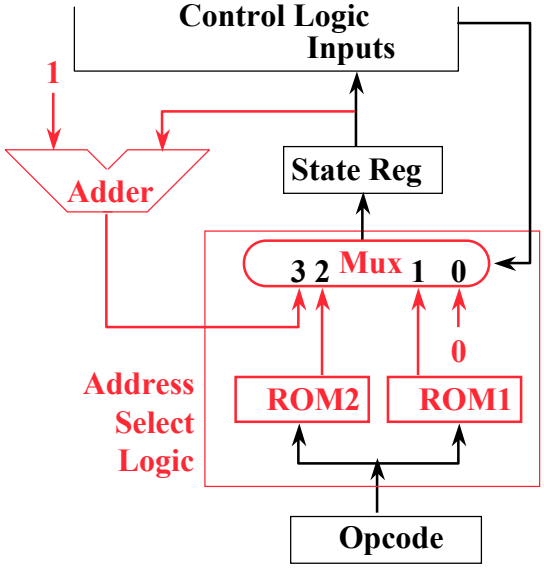


Sequencer-based control unit



361 multicontroller.19

Sequencer-based control unit details



Dispatch ROM 1

Op	*Name*	*State*
000000	Rtype	0110
000010	jmp	1001
000100	beq	1000
001011	ori	
1010 100011		
lw	0010	
101011	sw	0010

Dispatch ROM 2

Op	*Name*	*State*
100011	lw	0011
101011	sw	0101

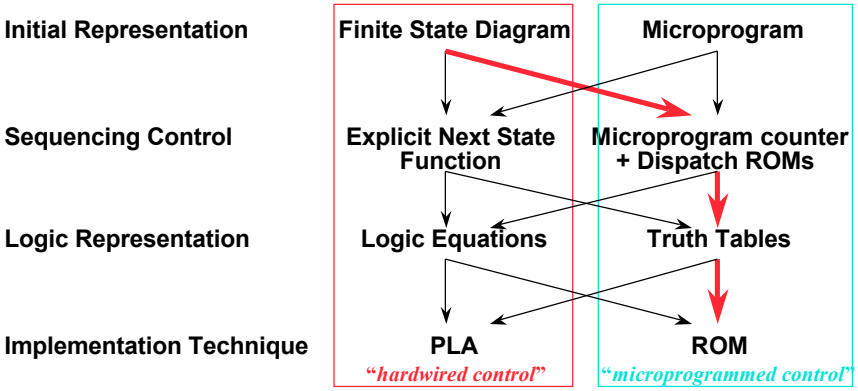
361 multicontroller.20

Implementing Control with a ROM

° Instead of a PLA, use a ROM with one word per state ("Control word")

State number	*Control Word Bits 18-2*	*Control Word Bits 1-0*
0	10010100000001000	11
1	00000000010011000	01
2	00000000000010100	10
3	00110000000010100	11
4	00110010000010110	00
5	00101000000010100	00
6	00000000001000100	11
7	00000000001000111	00
8	01000000100100100	00
9	10000001000000000	00
10	…	11
11	…	00

Next Iteration: Using Microprogram for Representation



° ROM can be thought of as a sequence of control words

° Control word can be thought of as instruction: "microinstruction"

° Rather than program in binary, use assembly language

Microprogramming

° Control is the hard part of processor design
  ° Datapath is fairly regular and well-organized
  ° Memory is highly regular
  ° Control is irregular and global

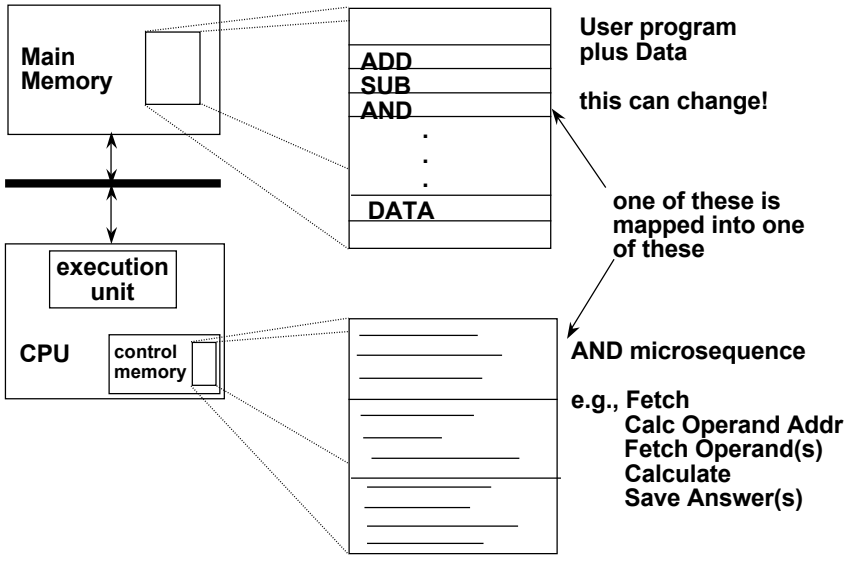
Microprogramming:

-- A Particular Strategy for Implementing the Control Unit of a processor by "programming" at the level of register transfer operations

Microarchitecture:

-- Logical structure and functional capabilities of the hardware as seen by the microprogrammer

Macroinstruction Interpretation



Microprogramming Pros and Cons

- Ease of design
- Flexibility
  - Easy to adapt to changes in organization, timing, technology
  - Can make changes late in design cycle, or even in the field
- Can implement very powerful instruction sets (just more control memory)
- Generality
  - Can implement multiple instruction sets on same machine.
  - Can tailor instruction set to application.
- Compatibility
  - Many organizations, same instruction set
- Costly to implement
- Slow

Summary: Multicycle Control

- Microprogramming and hardwired control have many similarities, perhaps biggest difference is initial representation and ease of change of implementation, with ROM generally being easier than PLA