

EECS 370

EECS 370 – Introduction to Computer Organization – Winter 2009

Prof. Todd Austin & Prof. Marios Papaefthymiou

EECS Department
University of Michigan in Ann Arbor, USA

© T. Austin & M. Papaefthymiou, 2009

The material in this presentation cannot be
copied in any form without our written permission

Today's lecture

- Multi-cycle data path
 - Improve processor performance
 - Increase clock speed
 - Reduce the combinational circuit complexity

What's Wrong with Single Cycle?

- 1 ns – Register read/write time
- 2 ns – ALU/adder
- 2 ns – memory access
- 0 ns – MUX, PC access, sign extend, ROM

	Get Instr	read reg	ALU operation	mem	write reg						
add:	2ns	+	1ns	+	2ns		+	1ns	=	6 ns	
beq:	2ns	+	1ns	+	2ns				=	5 ns	
sw:	2ns	+	1ns	+	2ns	+	2ns		=	7 ns	
lw:	2ns	+	1ns	+	2ns	+	2ns	+	1ns	=	8 ns

The University of Michigan
© T. Austin & M. Papaefthymiou – 2009

Computing Execution Time

Assume: 100 instructions executed

- 25% of instructions are loads,
- 10% of instructions are stores,
- 45% of instructions are adds, and
- 20% of instructions are branches.

Single-cycle execution:

$$100 * 8\text{ns} = \underline{800} \text{ ns}$$

Optimal execution:

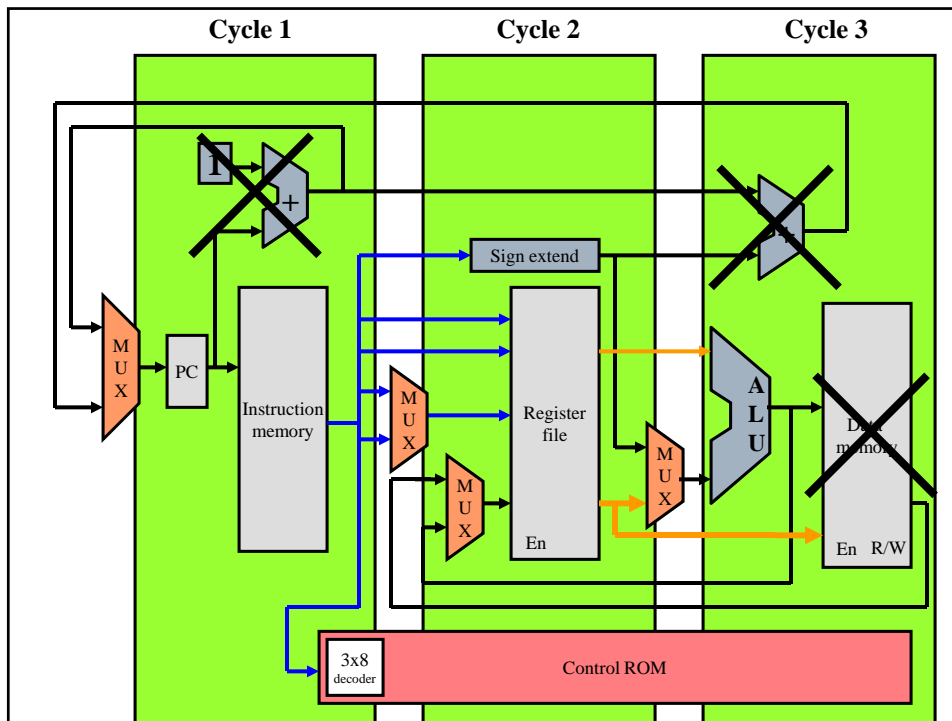
$$25*8\text{ns} + 10*7\text{ns} + 45*6\text{ns} + 20*5\text{ns} = \underline{640} \text{ ns}$$

The University of Michigan
© T. Austin & M. Papaefthymiou – 2009

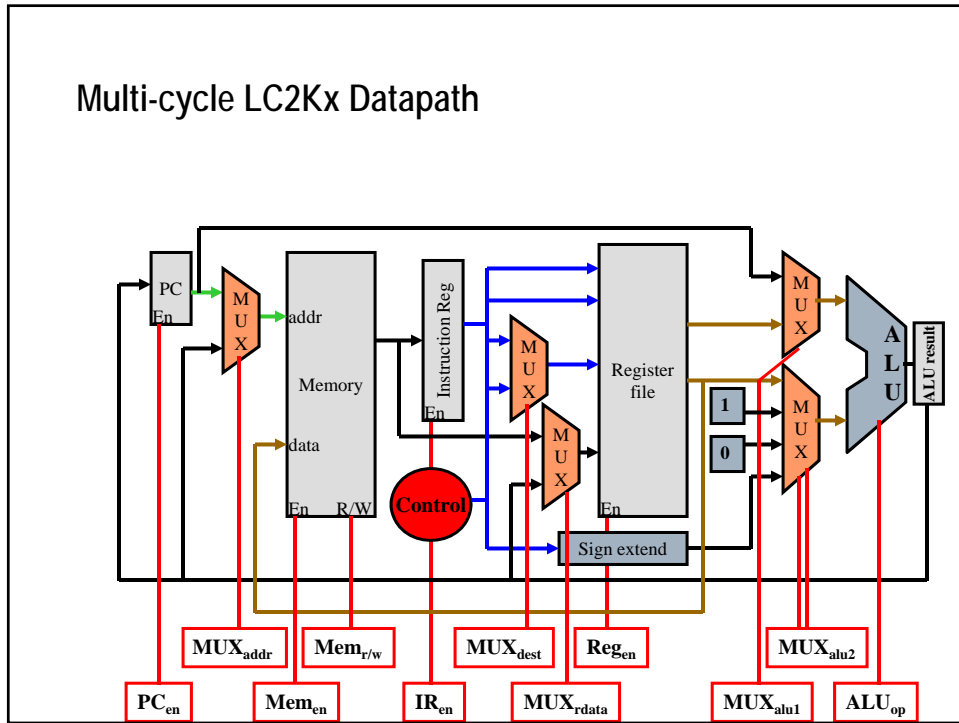
Multiple-cycle Execution

- ❑ Each instruction takes multiple cycles to execute
 - Cycle time is reduced
 - Slower instructions take more cycles
 - Can reuse datapath elements each cycle
- ❑ What is needed to make this work?
 - Since you are re-using elements for different purposes, you need more and/or wider MUXes.
 - You may need extra registers if you need to remember an output for 1 or more cycles.
 - Control is more complicated since you need to send new signals on each cycle.

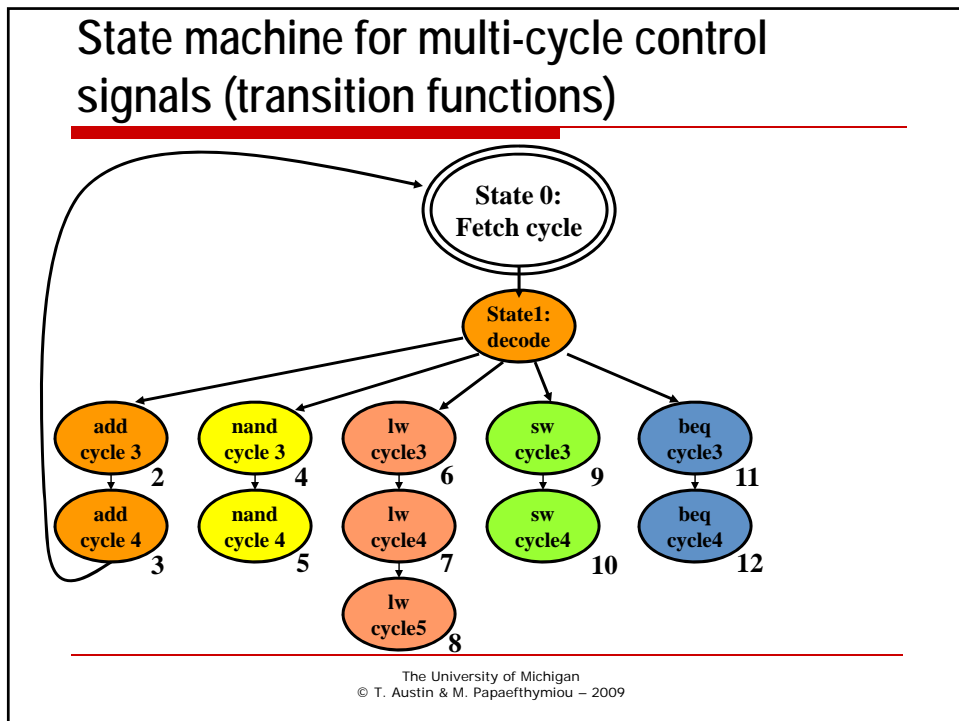
The University of Michigan
© T. Austin & M. Papaefthymiou – 2009



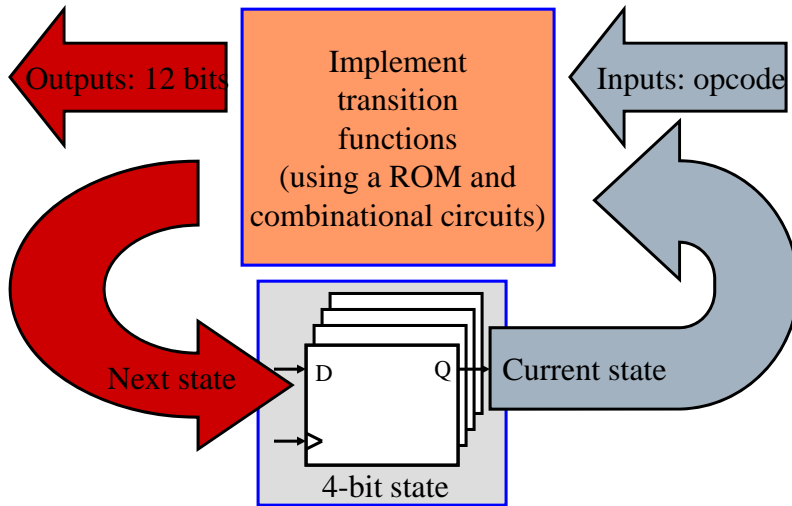
Multi-cycle LC2Kx Datapath



State machine for multi-cycle control signals (transition functions)

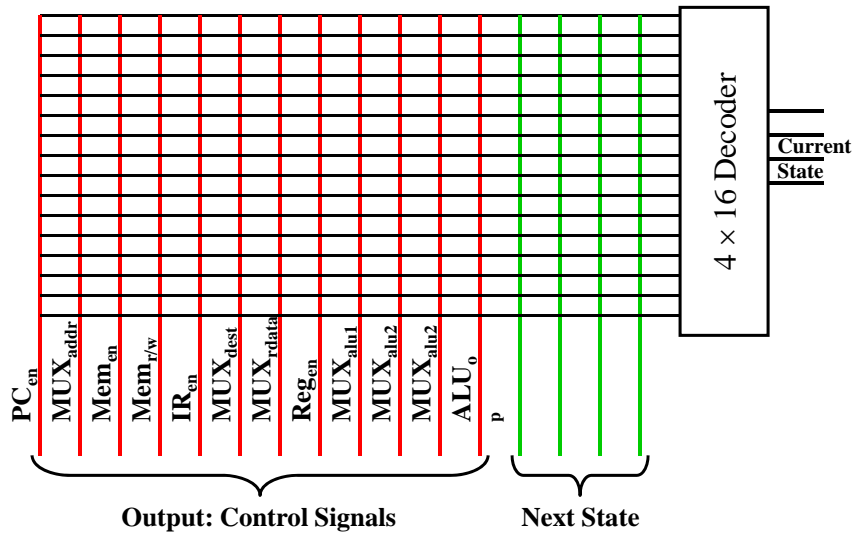


Implementing FSM



The University of Michigan
© T. Austin & M. Papaefthymiou - 2009

Building the Control Rom



First Cycle (State 0) Fetch Instr

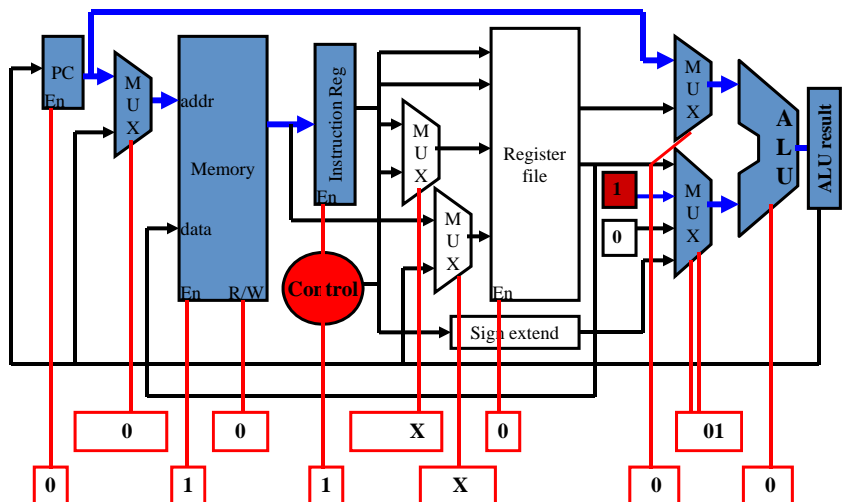
- ❑ What operations need to be done in the first cycle of executing any instruction?
 - Read memory[PC] and store into instruction register.
 - Must select PC in memory address MUX ($MUX_{addr} = 0$)
 - Enable memory operation ($Mem_{en} = 1$)
 - R/W should be (read) ($Mem_{r/w} = 0$)
 - Enable Instruction Register write ($IR_{en} = 1$)
 - Calculate PC + 1
 - Send PC to ALU ($MUX_{alu1} = 0$)
 - Send 1 to ALU ($MUX_{alu2} = 01$)
 - Select ALU add operation ($ALU_{op} = 0$)
 - $Pc_{en} = 0$; $Reg_{en} = 0$; MUX_{dest} and $MUX_{rddata} = X$

❑ Next State: Decode Instruction

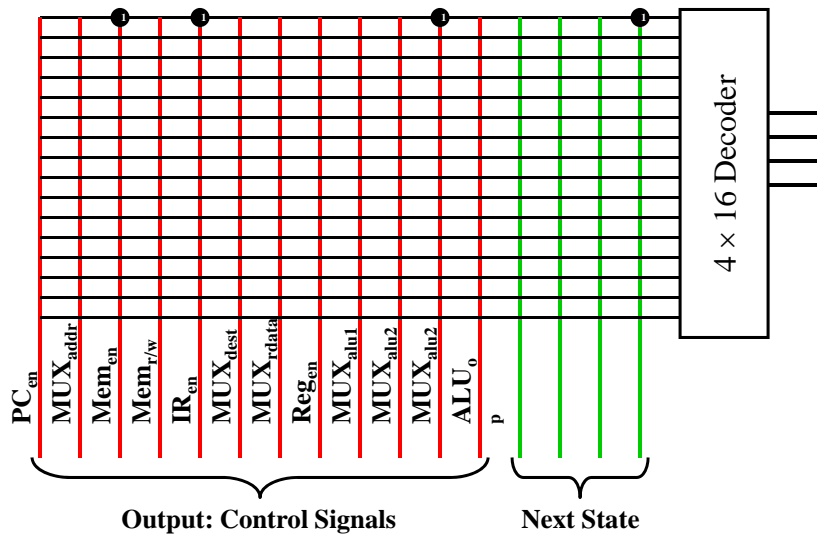
The University of Michigan
© T. Austin & M. Papaefthymiou – 2009

Cycle 1 Operation

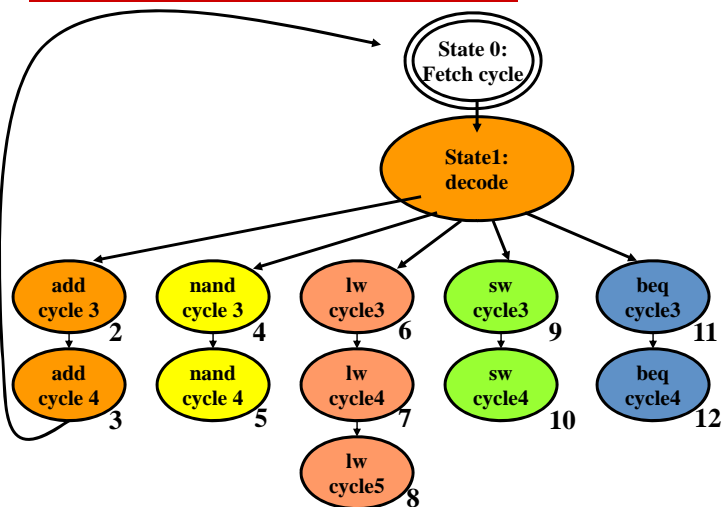
This is the same for all instructions
(since we don't know the instruction yet!)



Building the Control Rom



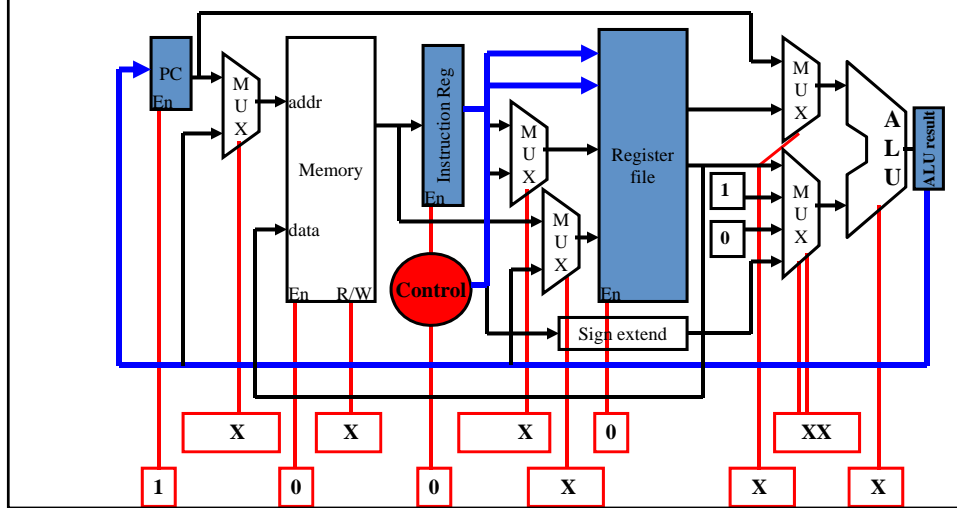
State 1: instruction decode



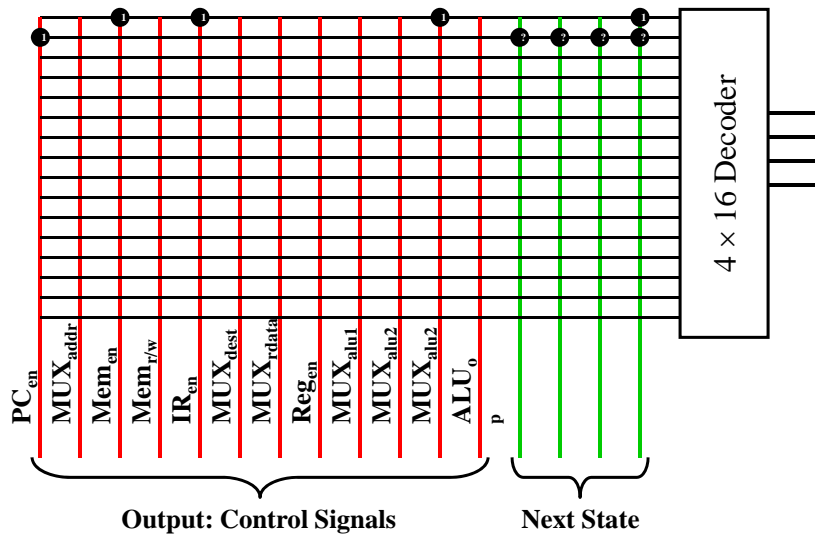
The University of Michigan
© T. Austin & M. Papaefthymiou – 2009

State 1: output function

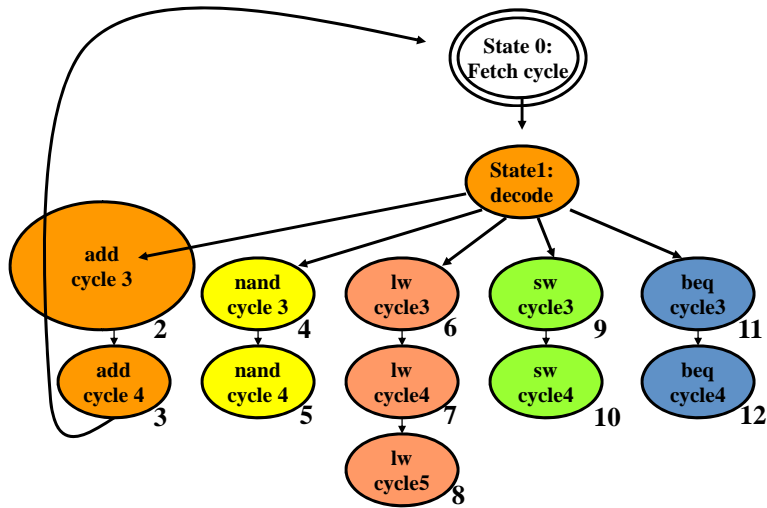
Update PC; read registers (RegA and regB);
use opcode to determine next state



Building the Control Rom



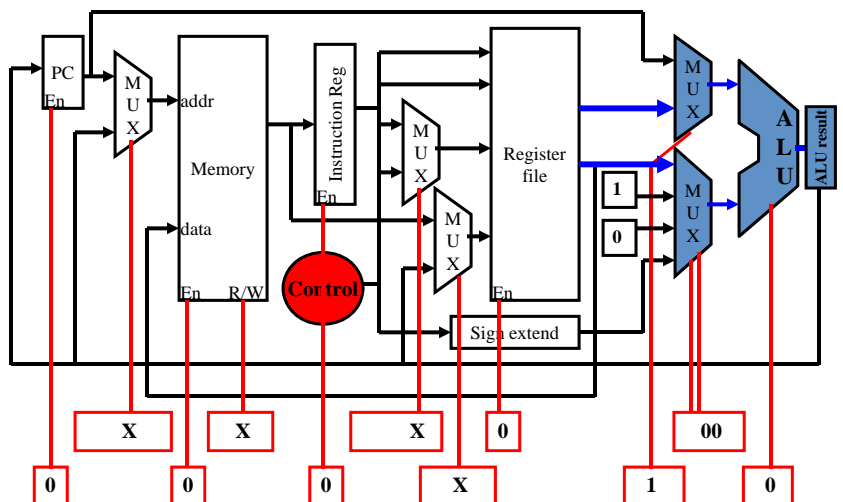
State 2: Add cycle 3



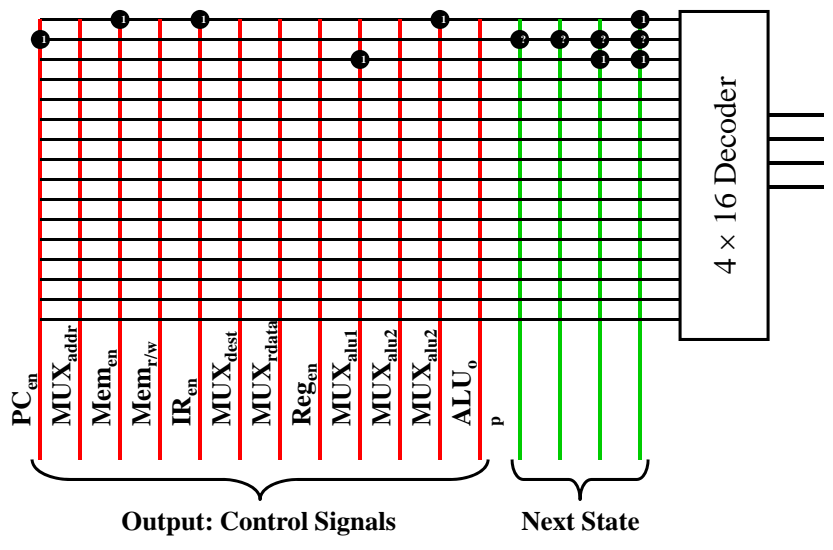
The University of Michigan
© T. Austin & M. Papaefthymiou – 2009

State 2: Add Cycle 3 Operation

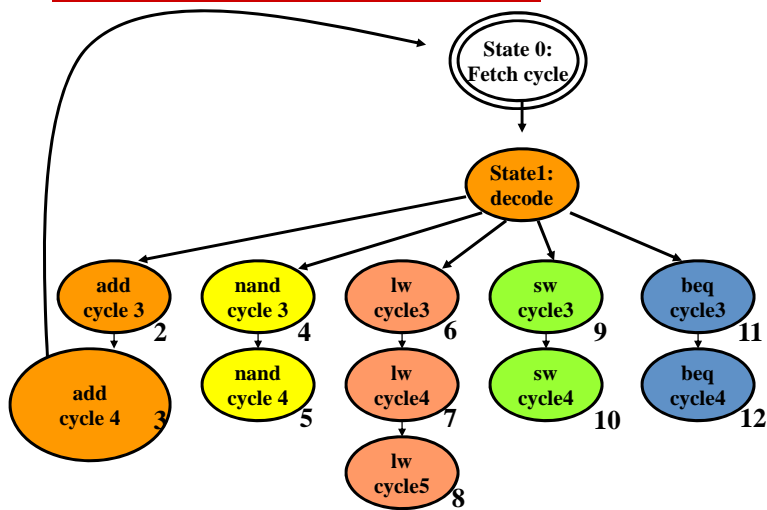
Send control signals to MUX to select values of regA and regB and control signal to ALU to add



Building the Control Rom



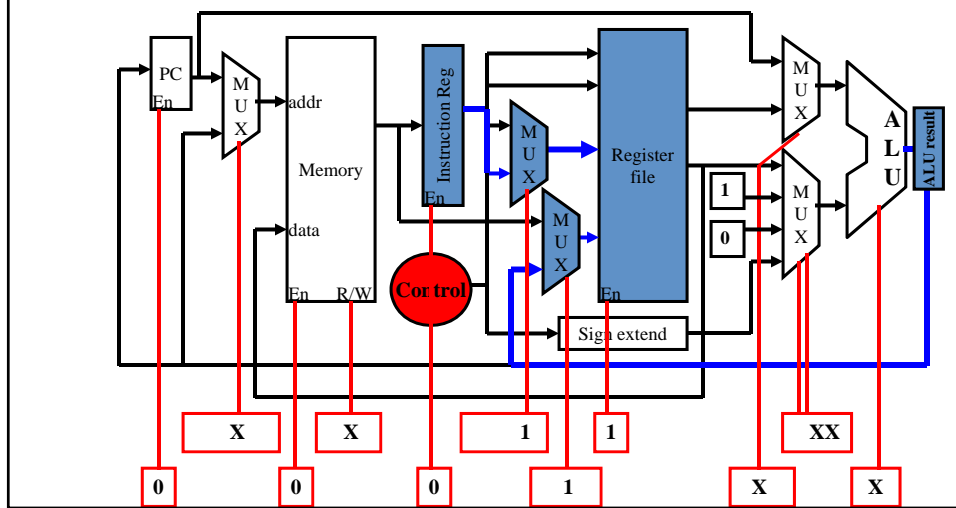
State 2: Add cycle 3



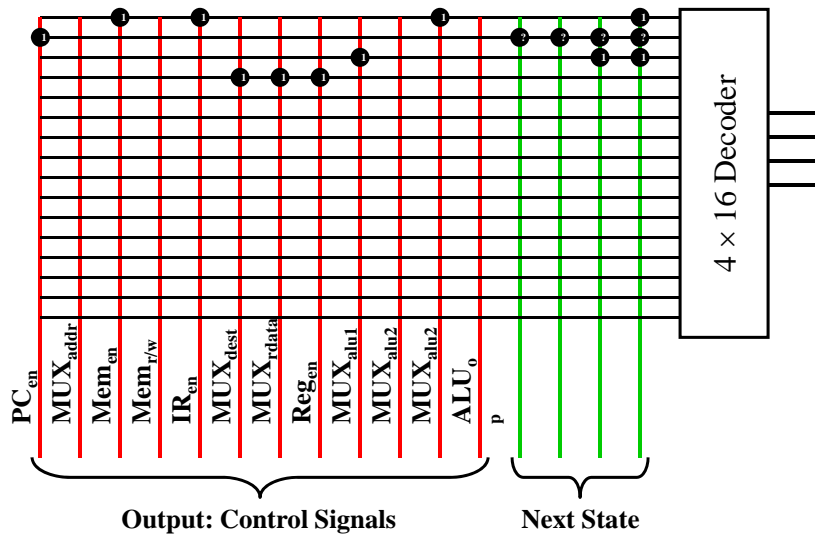
The University of Michigan
© T. Austin & M. Papaefthymiou – 2009

Add Cycle 4 Operation

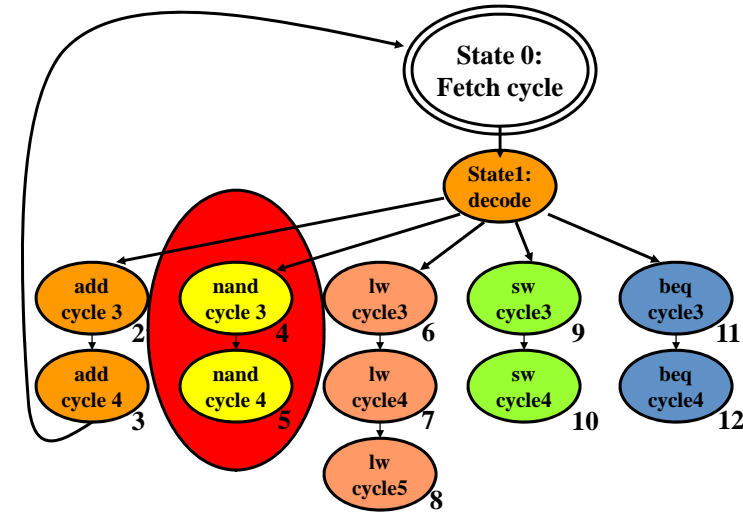
Send control signal to address MUX to select dest and to data MUX to select ALU output, then send write enable to register file.



Building the Control Rom

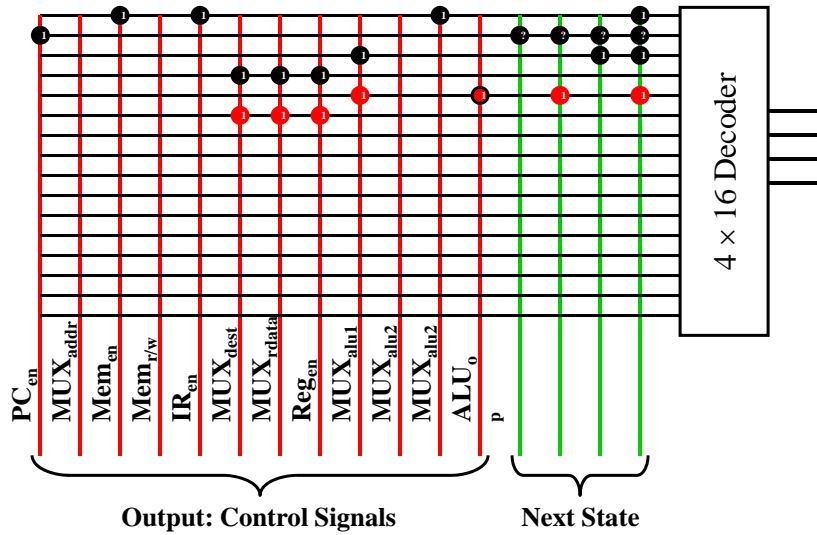


Return to State 0: Fetch cycle to execute the next instruction

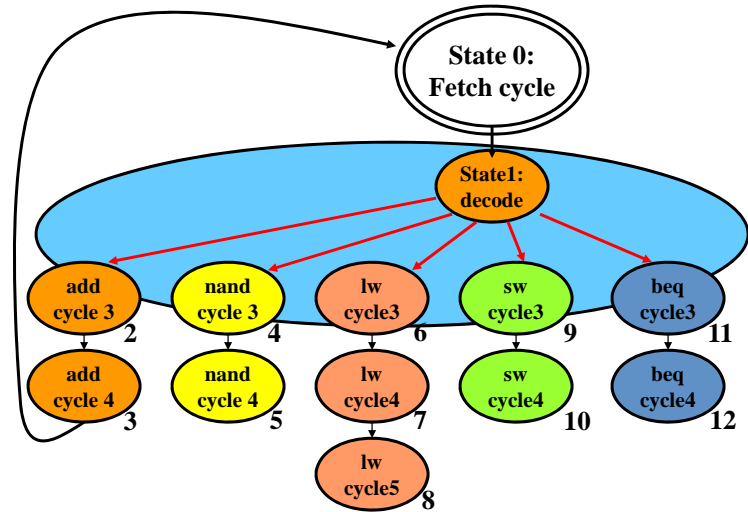


The University of Michigan
© T. Austin & M. Papaefthymiou - 2009

Control Rom for nand (4 and 5)

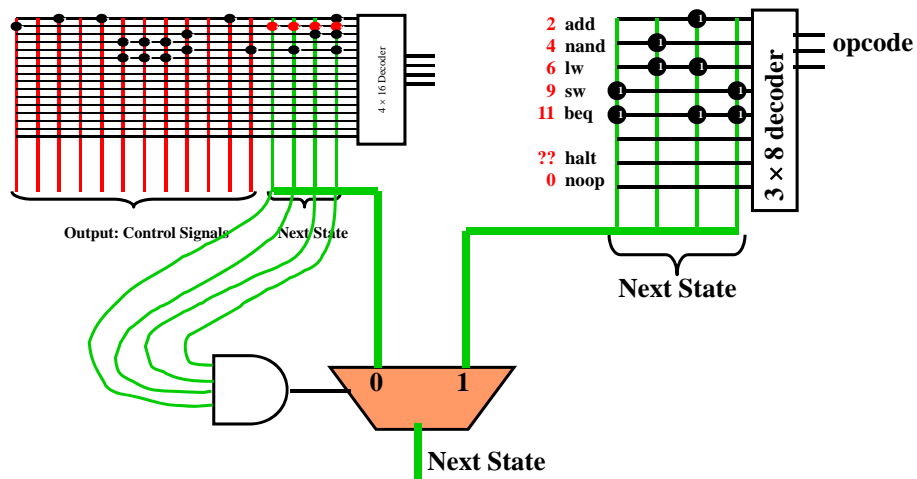


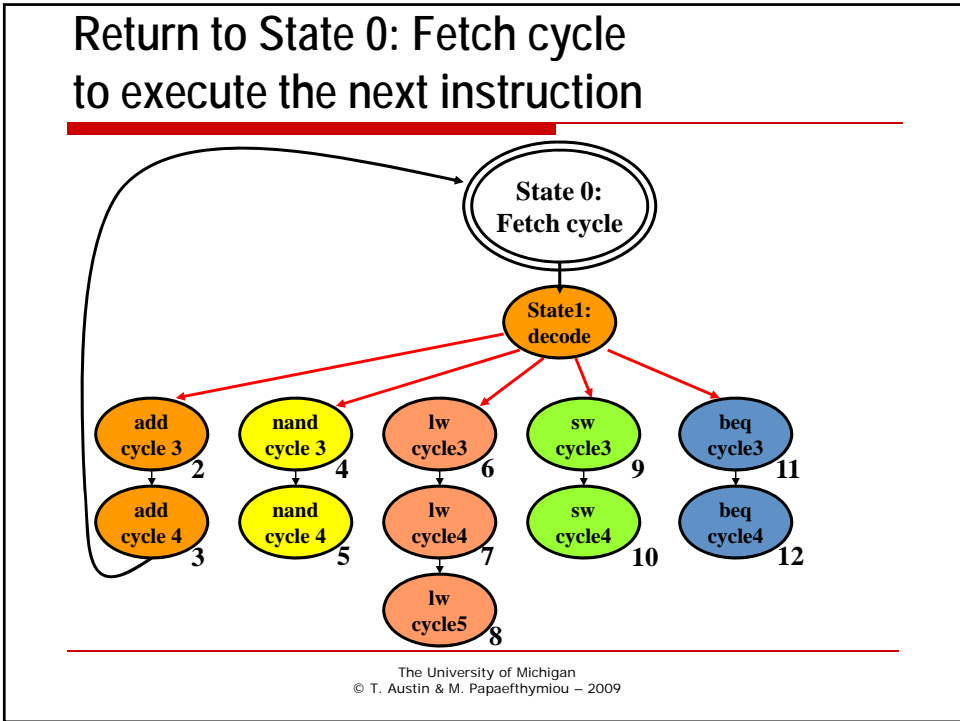
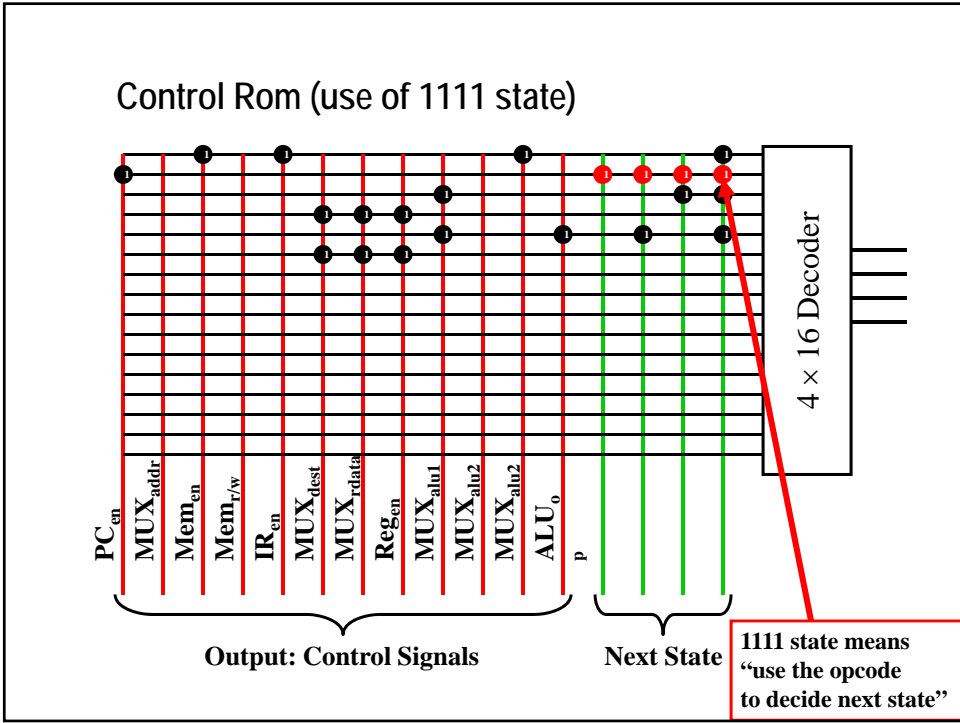
What about the transition from state 1?



The University of Michigan
© T. Austin & M. Papaefthymiou - 2009

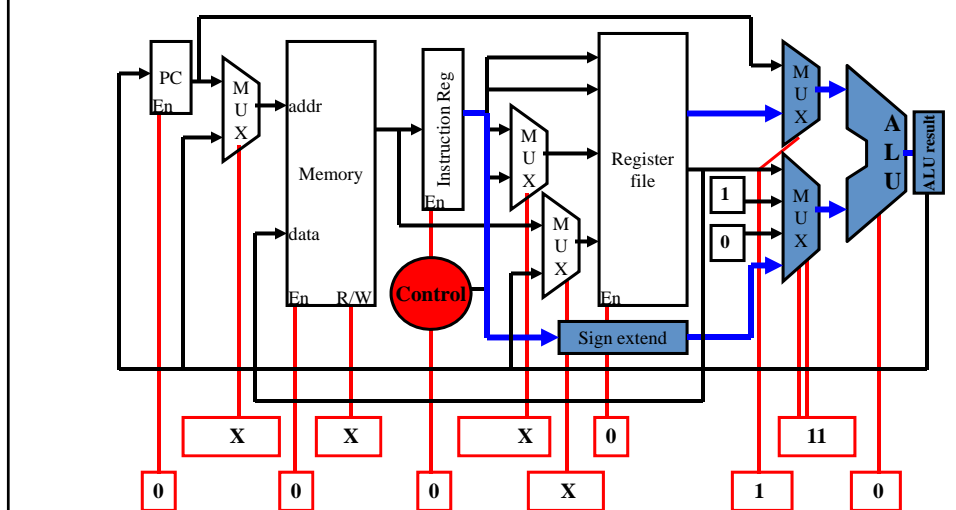
Complete transition function circuit



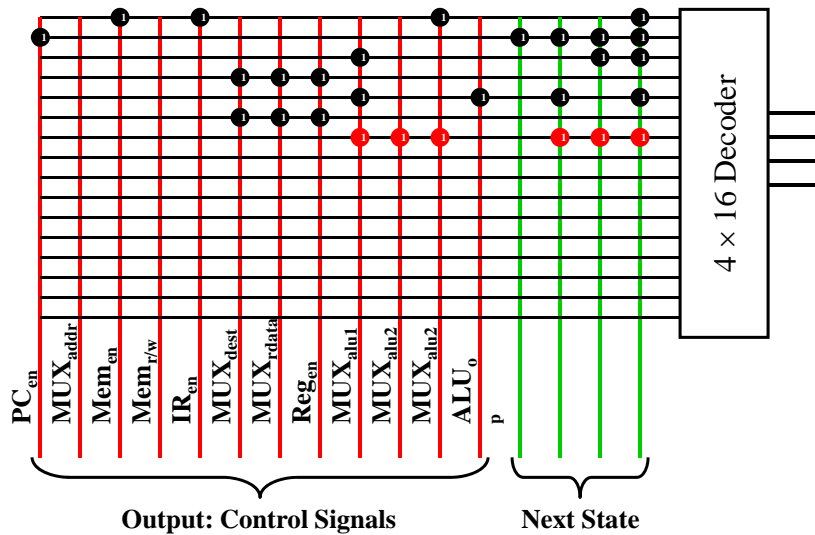


State 6: LW cycle 3

Calculate address for memory reference

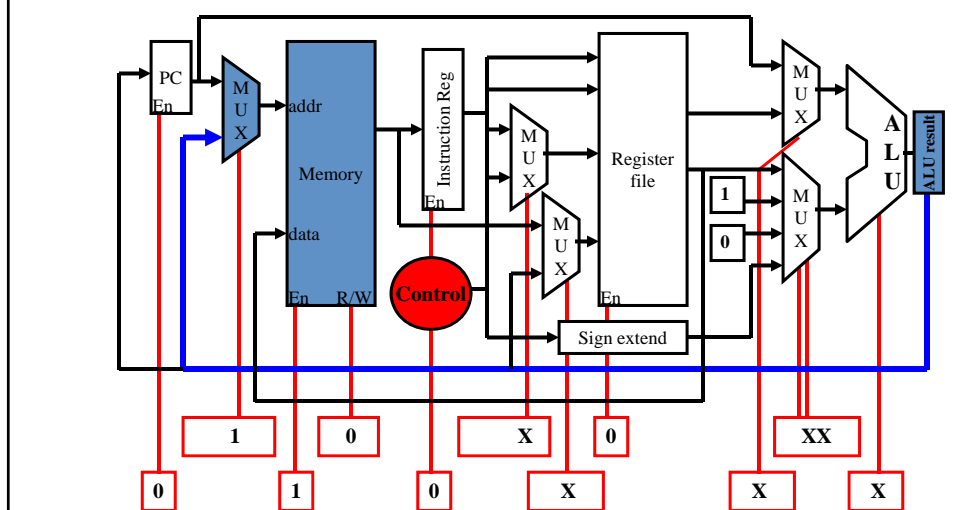


Control Rom (lw cycle 3)

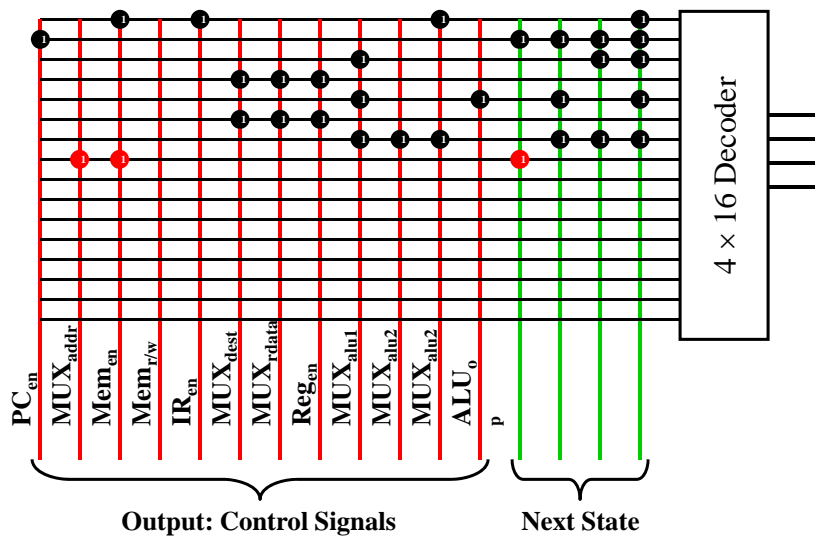


State 7: LW cycle 4

Read memory location

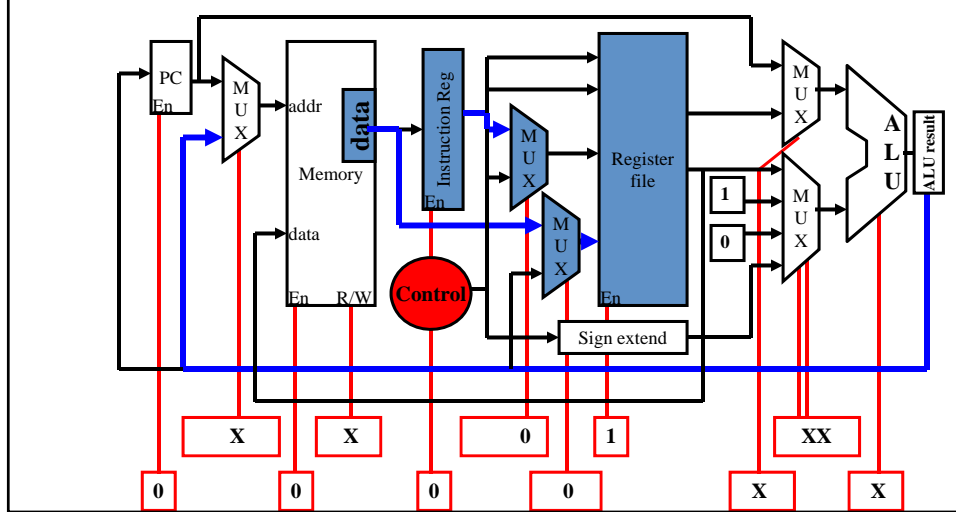


Control Rom (lw cycle 4)

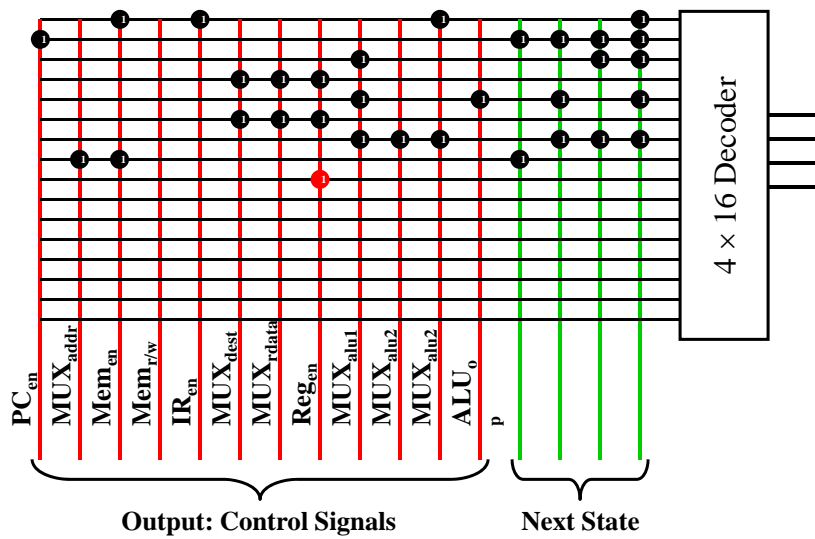


State 8: LW cycle 5

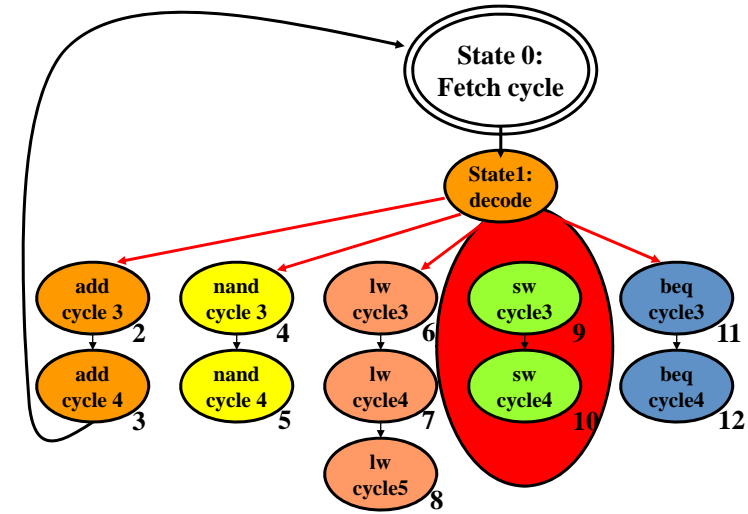
Write memory value to register file



Control Rom (lw cycle 5)

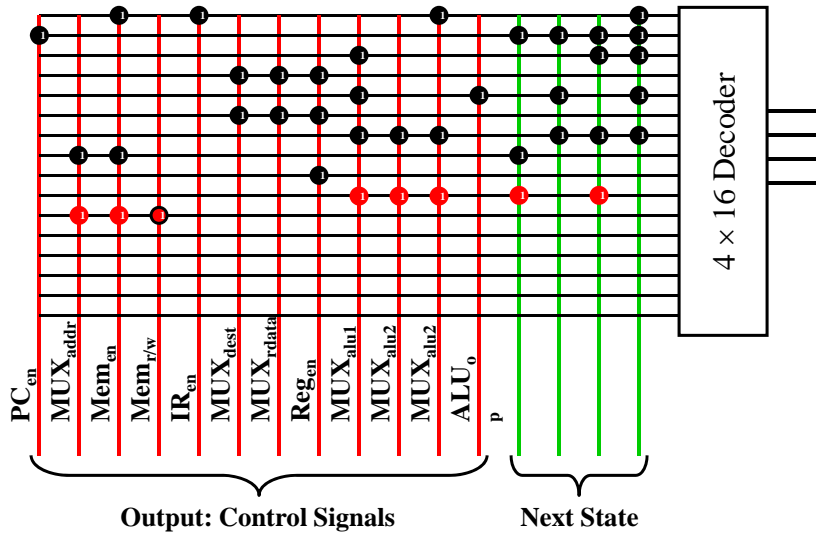


Return to State 0: Fetch cycle to execute the next instruction

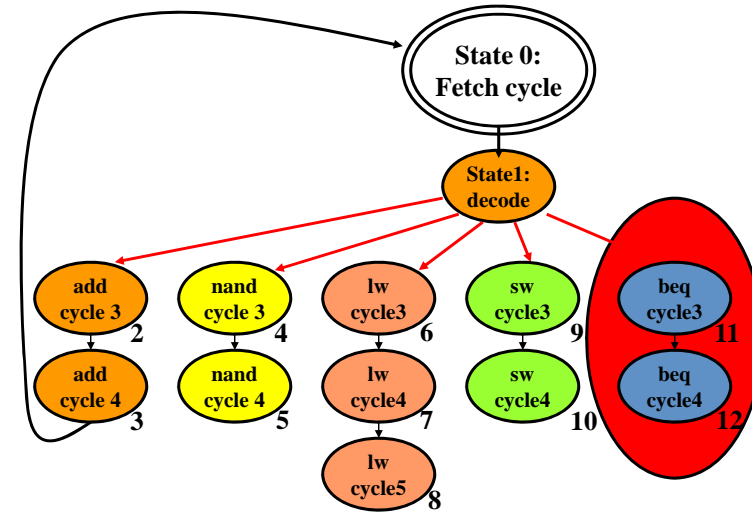


The University of Michigan
© T. Austin & M. Papaefthymiou - 2009

Control Rom (sw cycles 3 and 4)



Return to State 0: Fetch cycle to execute the next instruction



The University of Michigan
© T. Austin & M. Papaefthymiou - 2009

State 11: beq cycle 3

Calculate target address for branch

