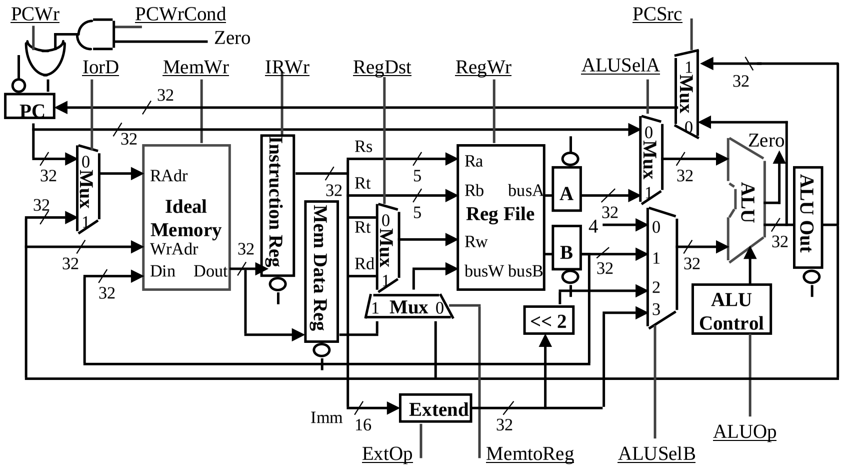
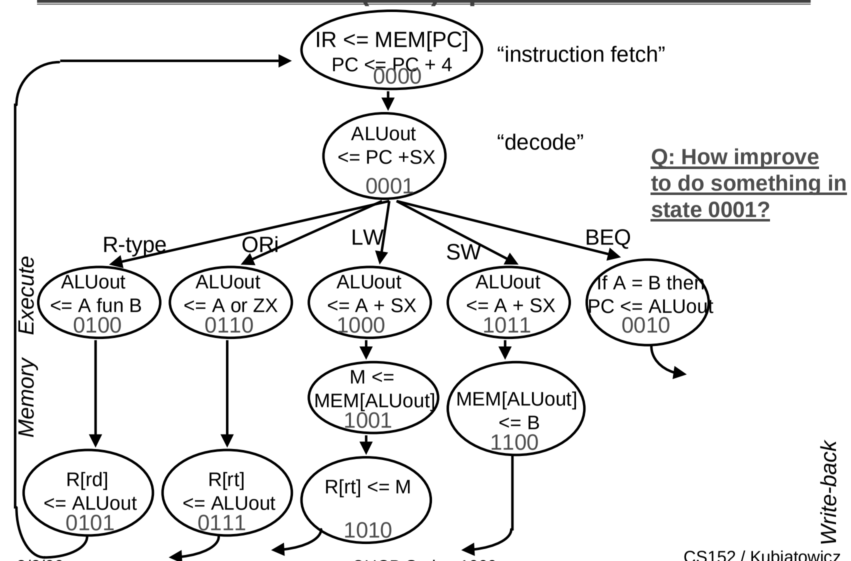
Alternative datapath (book): Multiple Cycle Datapath

° **Miminizes Hardware: 1 memory, 1 adder**



Finite State Machine (FSM) Spec



Q: How improve to do something in state 0001?

4) Legend of Fields and Symbolic Names

Field Name	***Values for Field***	***Function of Field with Specific Value***
ALU	**Add**	**ALU adds**
	Subt.	**ALU subtracts**
	Func code	**ALU does function code**
	Or	**ALU does logical OR**
SRC1	**PC**	**1st ALU input = PC**
	rs	**1st ALU input = Reg[rs]**
SRC2	**4**	**2nd ALU input = 4**
	Extend	**2nd ALU input = sign ext. IR[15-0]**
	Extend0	**2nd ALU input = zero ext. IR[15-0]**
	Extshft	**2nd ALU input = sign ex., sl IR[15-0]**
	rt	**2nd ALU input = Reg[rt]**
destination	**rd ALU**	**Reg[rd] = ALUout**
	rt ALU	**Reg[rt] = ALUout**
	rt Mem	**Reg[rt] = Mem**
Memory	**Read PC**	**Read memory using PC**
	Read ALU	**Read memory using ALU output**
	Write ALU	**Write memory using ALU output**
Memory register	**IR**	**IR = Mem**
PC write	**ALU**	**PC = ALU**
	ALUoutCond	**IF ALU Zero then PC = ALUout**
Sequencing	**Seq**	**Go to sequential µinstruction**
	Fetch	**Go to the first microinstruction**
	Dispatch	**Dispatch using ROM.**

Microprogram it yourself!

Label	*ALU*	*SRC1*	*SRC2*	*ALU Dest.*	*Memory*	*Mem. Reg.*	*PC Write*	*Sequencing*
Fetch:	**Add**	**PC**	**4**		**Read PC**	**IR**	**ALU**	**Seq**

1&2) Start with list of control signals, grouped into fields

Single Bit Control

Signal name	Effect when deasserted	Effect when asserted
ALUSelA	1st ALU operand = PC	1st ALU operand = Reg[rs]
RegWrite	None	Reg. is written
MemtoReg	Reg. write data input = ALU	Reg. write data input = memory
RegDst	Reg. dest. no. = rt	Reg. dest. no. = rd
MemRead	None	Memory at address is read, MDR <= Mem[addr]
MemWrite	None	Memory at address is written
IorD	Memory address = PC	Memory address = S
IRWrite	None	IR <= Memory
PCWrite	None	PC <= PCSource
PCWriteCond	None	IF ALUzero then PC <= PCSource
PCSource	PCSource = ALU	PCSource = ALUout

Multiple Bit Control

Signal name	Value	Effect
ALUOp	00	ALU adds
	01	ALU subtracts
	10	ALU does function code
	11	ALU does logical OR
ALUSelB	000	2nd ALU input = Reg[rt]
	001	2nd ALU input = 4
	010	2nd ALU input = sign extended IR[15-0]
	011	2nd ALU input = sign extended, shift left 2 IR[15-0]
	100	2nd ALU input = zero extended IR[15-0]