Microprogramming and Exceptions

3-26-2003

Opening Discussion

- What did we talk about last class?
- Have you seen anything interesting in the news?
- What is the FSM that generates or recognizes the language $(10)^*$?

Microinstruction Fields

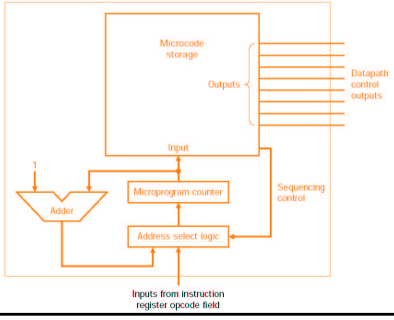
- The main trick of designing the microinstructions is defining fields that control non-overlapping sets of control lines.

Field	Values
Label	Any String
ALU control	Add, Subt, Func code
SRC1	PC, A
SRC2	B, 4, Extend, ExtShft
Register control	Read, Write ALU, Write MDR
Memory	Read PC, Read ALU, Write ALU
PCWrite control	ALU, ALUOut-cond, Jump address
Sequencing	Seq, Fetch, Dispatch i

Our Microprogram

Label	ALU Co	SRC1	SRC2	Reg Co	Memory	PCWrite	Seq
Fetch	Add	PC	4		Read PC	ALU	Seq
	Add	PC	Extshft	Read			Dispatch 1
Mem1	Add	A	Extend				Dispatch 2
LW2					Read ALU		Seq
				Write MDR			Fetch
SW2					Write ALU		Fetch
Rformat1	Func code	A	B				Seq
				Write ALU			Fetch
BEQ1	Subt	A	B			ALUOut-cond	Fetch
JUMP1						Jump address	Fetch

Implementing the Microprogram



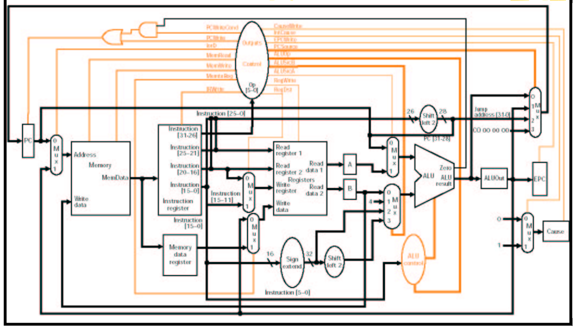
Exceptions

- We saw exceptions before with overflows in numerical operations. The term exception is used for MIPS to describe an abnormal change in program flow caused by something in the processor. Interrupts come from external sources. Many platforms use interrupt for both of these.
- The primary outside source is I/O which is in chapter 8.

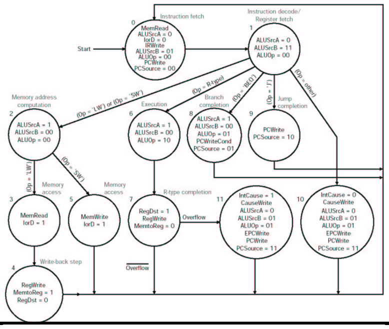
Details of Exceptions

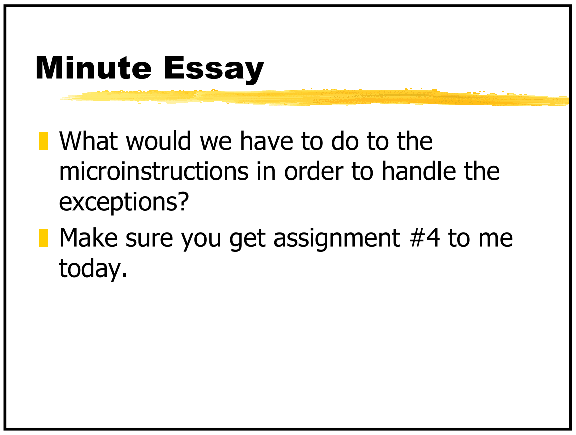
- With what we know, there are two possible exceptions: arithmetic overflow and undefined instruction.
- The PC value where the exception occurred is stored in the EPC. The "cause register" holds a values that tells us what the cause was.

Multicycle Datapath with Exceptions



FSM with Exceptions





Minute Essay

- What would we have to do to the microinstructions in order to handle the exceptions?
- Make sure you get assignment #4 to me today.