

UNIVERSITY OF MASSACHUSETTS
Dept. of Electrical & Computer Engineering

Computer Architecture
ECE 568

Part 2

Pipelining - 1

Israel Koren
Fall 2009

ECE568/Koren Part.2.1

Adapted from Patterson, Katz and Kubiawicz © UCB

Copyright 2005 UCB & Morgan Kaufmann

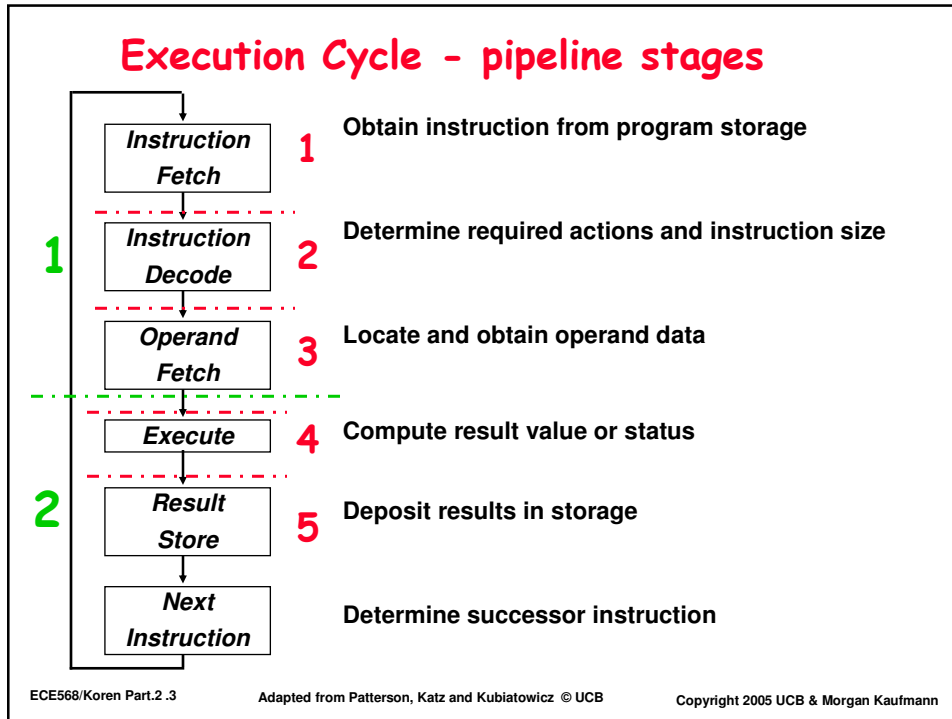
Instruction Execution - Pipelines

- ◆ Execute billions of instructions, so *throughput* is what matters
- ◆ What is desirable in instruction sets for pipelining?
 - Variable length instructions vs. all instructions same length?
 - Memory operands part of any operation vs. memory operands only in loads or stores?
 - Register operand in various places in instruction format vs. registers located in same place?
- ◆ Conclusion: RISC is easier to pipeline

ECE568/Koren Part.2.2

Adapted from Patterson, Katz and Kubiawicz © UCB

Copyright 2005 UCB & Morgan Kaufmann



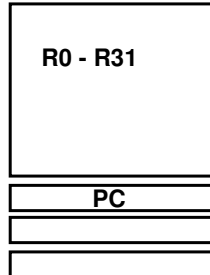
- ### "MIPS" - A "Typical" RISC
- ◆ 32-bit fixed length instruction (3 formats)
 - ◆ Memory access only via load/store instructions
 - ◆ 32 32-bit GPR (R0 contains zero, DP take pair)
 - or 32 64-bit GPR (integer reg) + 32 32-bit FPR (DP take pair)
 - ◆ 3-address, reg-reg arithmetic instruction; registers in same place in instruction format
 - ◆ Single address mode for load/store: base + displacement
 - no indirection
 - ◆ Simple branch conditions
 - ◆ Delayed branch
- see: some versions of SPARC, MIPS, HP PA-Risc, DEC Alpha, IBM PowerPC, DSP processors
pp. 129-136, A.26-A.37 in textbook
- ECE568/Koren Part.2.4 Adapted from Patterson, Katz and Kubiawicz © UCB Copyright 2005 UCB & Morgan Kaufmann

MIPS R3000 Instruction Set Architecture

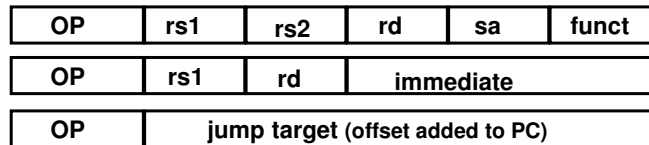
◆ Instruction Categories

- Load/Store
- Computational (Fixed-point etc)
- Floating-Point
- Jump and Branch
- Special

Registers



3 Instruction Formats: all 32 bits wide



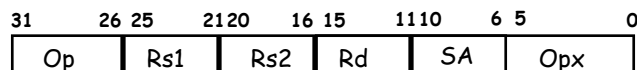
ECE568/Koren Part.2.5

Adapted from Patterson, Katz and Kubiawicz © UCB

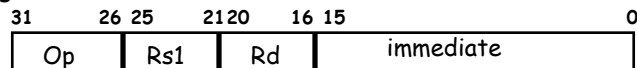
Copyright 2005 UCB & Morgan Kaufmann

MIPS Instruction layout

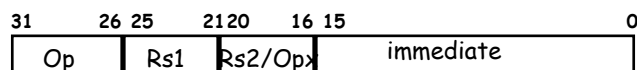
Register-Register



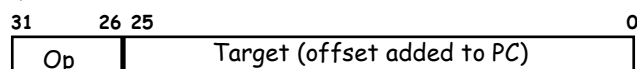
Register-Immediate



Branch



Jump / Call

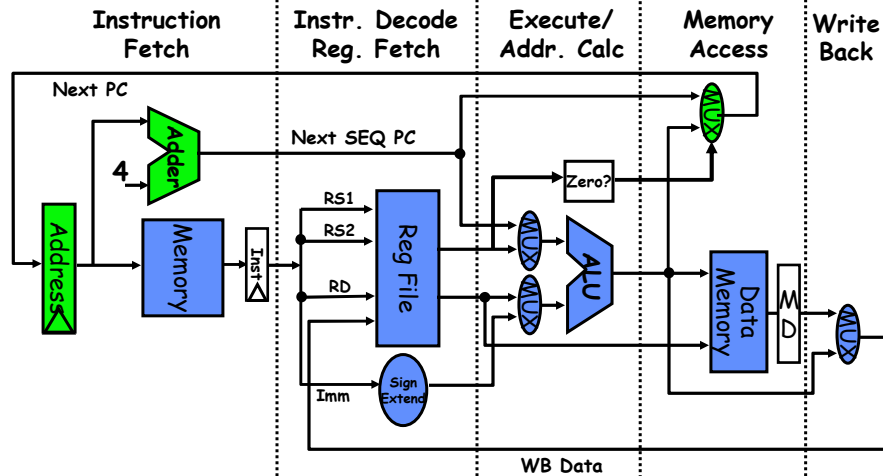


ECE568/Koren Part.2.6

Adapted from Patterson, Katz and Kubiawicz © UCB

Copyright 2005 UCB & Morgan Kaufmann

5 Steps of MIPS Datapath w/o pipelining



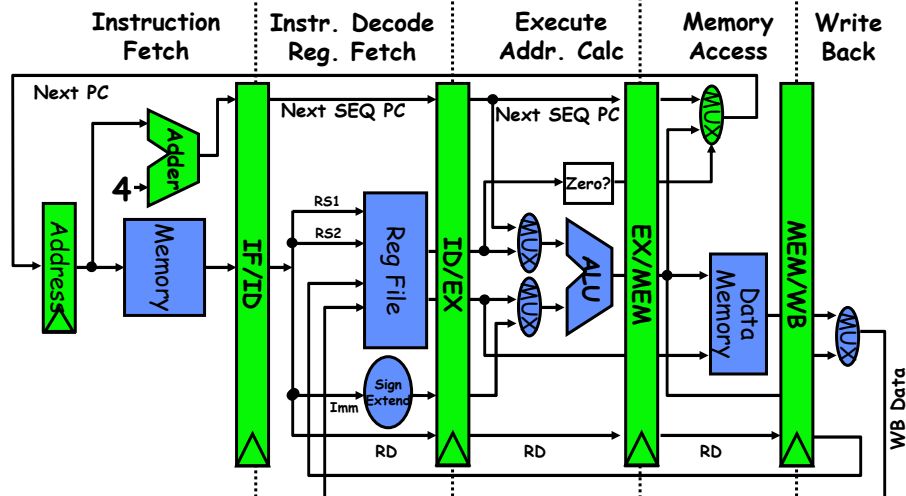
ECE568/Koren Part.2.7

Adapted from Patterson, Katz and Kubiawicz © UCB

Copyright 2005 UCB & Morgan Kaufmann

5 Steps of MIPS Datapath w/pipelining

Figure A.18, Page A-31, CA:AQA 3e



- Local decoded instruction fields for each phase / pipeline stage

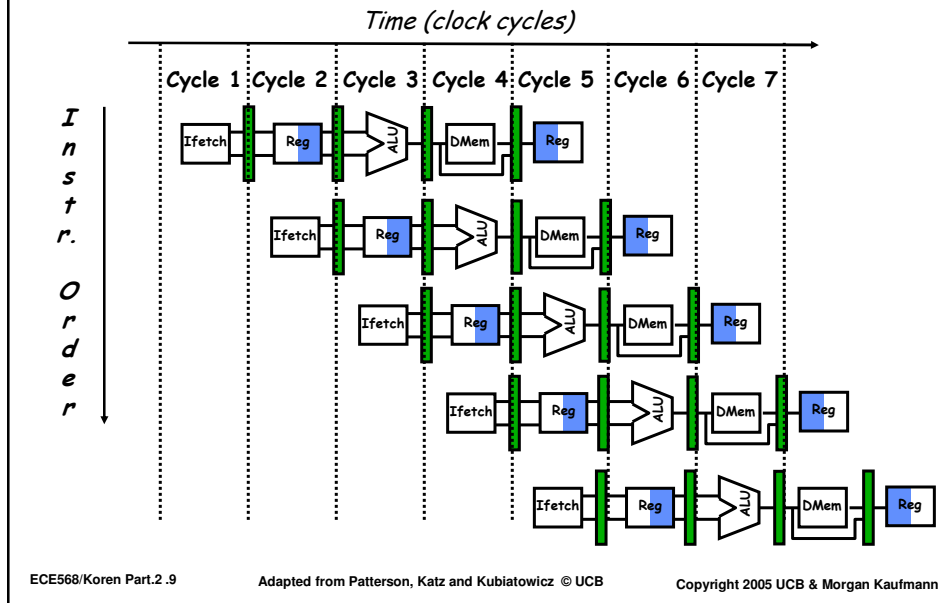
ECE568/Koren Part.2.8

Adapted from Patterson, Katz and Kubiawicz © UCB

Copyright 2005 UCB & Morgan Kaufmann

Visualizing Pipelining

Figure A.3, Page A-9, CA:AQA 3e



Instruction pipelines are not ideal

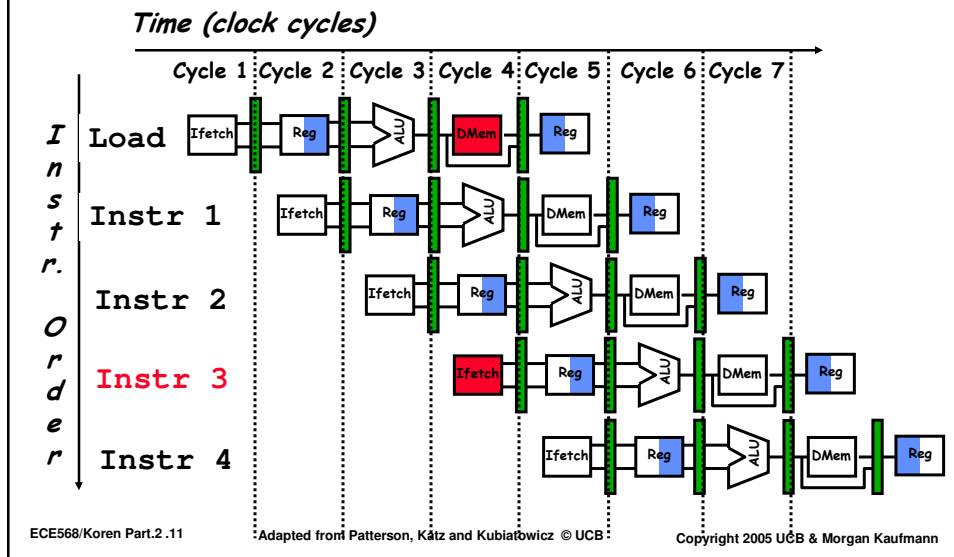
- ◆ Limits to pipelining: **Hazards** prevent next instruction from executing during its designated clock cycle
 - **Structural hazards**: Single HW resource needed by two instructions (e.g., operand address calculation using the same fixed-point adder used for addition)
 - **Data hazards**: Instruction depends on result of prior instruction still in the pipeline:

$$A \leftarrow B + C$$

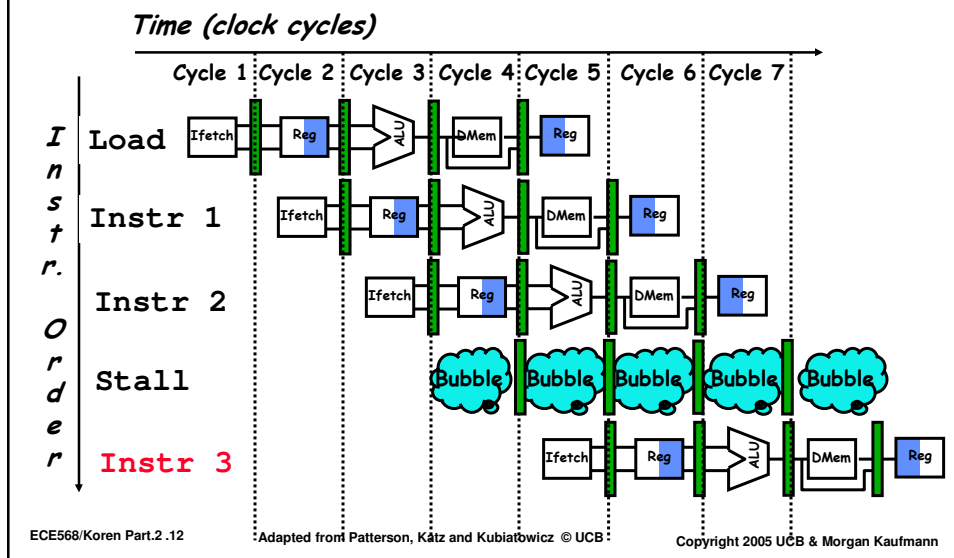
$$D \leftarrow A * B$$
 - **Control hazards**: Branches and jumps
 - **Interrupts/exceptions**
- ◆ **Issues**:
 - How to detect?
 - How to minimize the penalty?

Structural Hazards - one Memory Port

Figure A.4, Page A-14, CA:AQA 3e



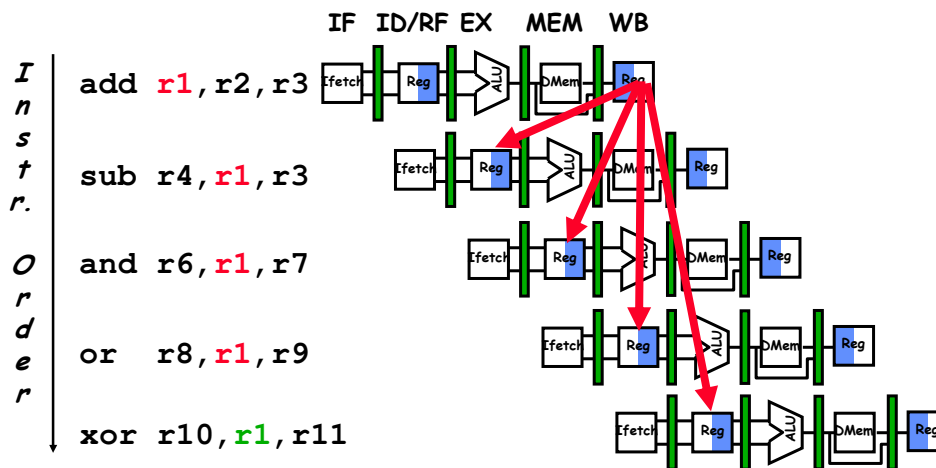
One Memory Port/Structural Hazards



Data Hazard on R1

Figure A.5, page A-17, CA:AQA 3e

Time (clock cycles)



ECE568/Koren Part.2.13

Adapted from Patterson, Katz and Kubiawicz © UCB

Copyright 2005 UCB & Morgan Kaufmann

Pipeline control design (avoid structural hazards)

“Effective control for pipelined computers,”
by Davidson *et al.* (available on the web page)

Reservation Table: Task A t0 t1 t2 t3 t4 t5

Unit s0 is the busiest

3 out of 6 time units

throughput ≤ 2 instr./ 6 cycles

$1/6 \leq$ throughput $\leq 1/3$

How can we find the real throughput?

Define: **Pipeline Collision Vector (PCV)**

PCV = C0 C1 ... C{k-1}; k = # of cycles (t0, ..., t{k-1})

Ci=1 if a 2nd task initiated i cycles after the 1st will
result in a collision; Ci=0 otherwise.

s0	X		X		X	
s1		X				
s2		X				X
s3			X	X		
s4			X			X

ECE568/Koren Part.2.14

Copyright 2008 Koren UMass

PCV for Reservation Table A

t=1

	t0	t1	t2	t3	t4	t5
s0	X	X	X	X	X	X
s1		X	X			
s2		X	X		X	X
s3			X	X	X	
s4			X	X	X	X

0 1 2 3 4 5
Initial Value PCV=(1 1 1 1 1 0)

Throughput =

t=2

	t0	t1	t2	t3	t4	t5
s0	X		X	X	X	
s1		X	X			
s2		X	X	X	X	X
s3			X	X	X	
s4			X	X	X	X

t=3

	t0	t1	t2	t3	t4	t5
s0	X		X	X	X	X
s1		X		X		
s2		X		X	X	X
s3			X	X	X	
s4			X	X	X	X

ECE568/Koren Part.2.15

Copyright 2008 Koren UMass

Forbidden Latencies

A

	t0	t1	t2	t3	t4	t5
s0	X		X		X	
s1		X				
s2		X			X	
s3			X	X		
s4			X		X	

s0: 2,4

Overall=0,1,2,3,4

s1: 0 only

s2: 4

s3: 1

s4: 3

0 1 2 3 4 5
PCV=(1 1 1 1 1 0)

B

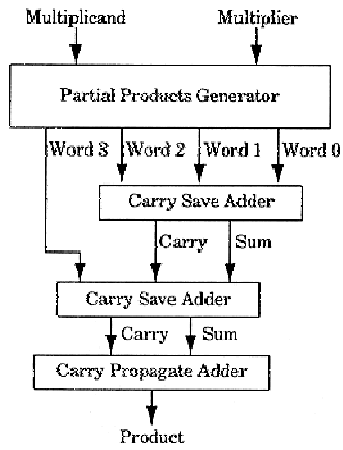
	t0	t1	t2	t3	t4	t5
s0	X	X				
s1	X		X			
s2	X			X		
s3	X				X	
s4	X					X

0 1 2 3 4 5
PCV(Table B)=()

ECE568/Koren Part.2.16

Copyright 2008 Koren UMass

Reservation Table for a multiplier



	1	2	3	4	5
PPG	X				
CSA1		X			
CSA2			X		
CPA				X	X

Source: "Memory Systems & Pipelined Processor," H. Cragon, Jones & Bartlett, 1996.

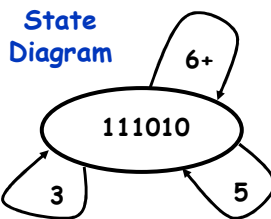
ECE568/Koren Part.2.17

Copyright 2008 Koren UMass

PCV for Reservation Table C

C

	t0	t1	t2	t3	t4	t5
s0	X		X		X	
s1		X				
s2			X	X		
s3		X				X



Forbidden latencies: 0,1,2,4

$$PCV(\text{Table } C) = (1 \ 1 \ 1 \ 0 \ 1 \ 0)$$

= state of pipeline at t0

At t1: 1 1 0 1 0 0

At t2: 1 0 1 0 0 0

At t3: 0 1 0 0 0 0

If a new task is started at t3:

0 1 0 0 0 0

1 1 1 0 1 0

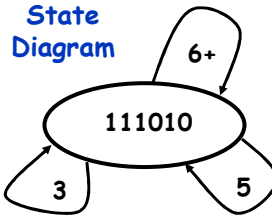
1 1 1 0 1 0

ECE568/Koren Part.2.18

Copyright 2008 Koren UMass

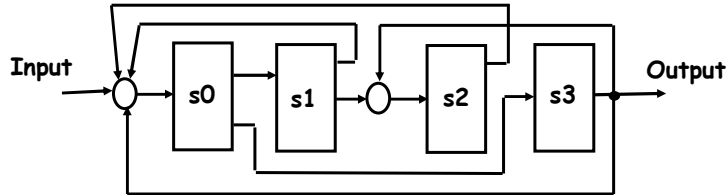
RT C - communication links

	t0	t1	t2	t3	t4	t5
s0	X		X		X	
s1		X				
s2			X	X		
s3		X				X



Possible execution cycles:
(3),(5),(3,5)

Necessary communication links (plus timing counter):



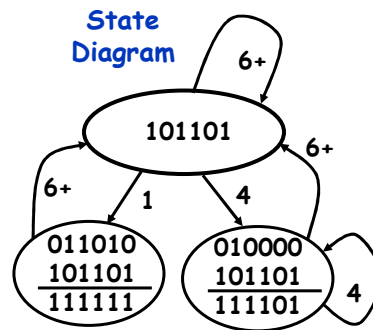
ECE568/Koren Part.2 :19

Copyright 2008 Koren UMass

Reservation Table D

	t0	t1	t2	t3	t4	t5
s0	X		X			X
s1		X				
s2		X	X			
s3					X	

Forbidden Latencies=?
PCV=?



Possible execution cycles: (4),(1,6),(4,6)

If (1,6) is followed: tasks start at 0,1,7,8,14,...

Avg.Delay=Avg.(1/throughput)=(1+6)/2=3.5

If (4) is followed: tasks start at 0,4,8,12,...

Avg.Delay=4

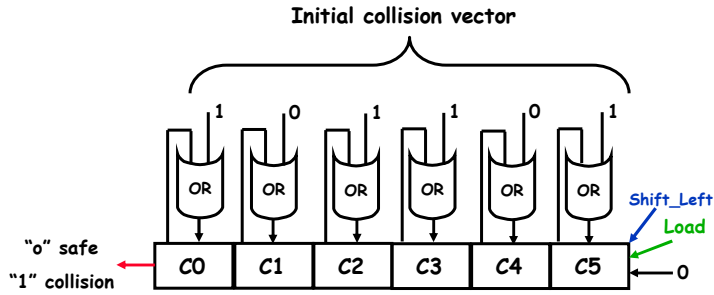
(1,6) - greedy cycle

Minimum Average Latency (MAL)=3.5; \leq MAL \leq

ECE568/Koren Part.2 :20

Copyright 2008 Koren UMass

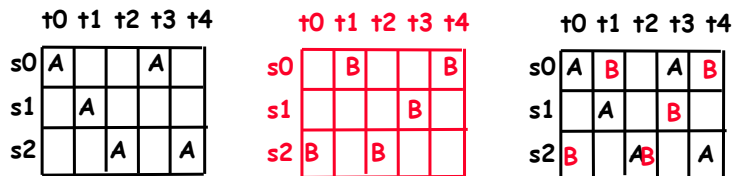
Greedy control



ECE568/Koren Part.2 .21

Copyright 2008 Koren UMass

Two tasks A & B

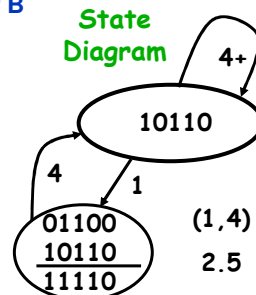


$$M_A = \begin{bmatrix} 01234 \\ 10110 \\ 10101 \end{bmatrix} \begin{matrix} A \text{ after A} \\ B \text{ After A} \end{matrix}$$

Forbidden latencies
B after A:
(from B to A)
0,2,4

$$M_B = \begin{bmatrix} 01234 \\ 11101 \\ 10110 \end{bmatrix} \begin{matrix} A \text{ after B} \\ B \text{ After B} \end{matrix}$$

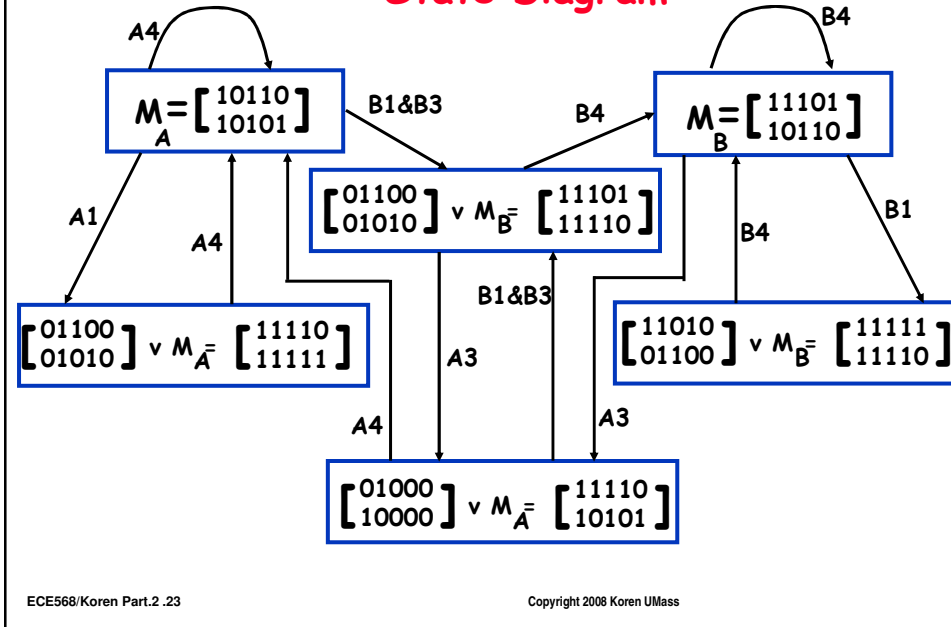
Forbidden latencies
A after B:
(from A to B)
0,1,2,4



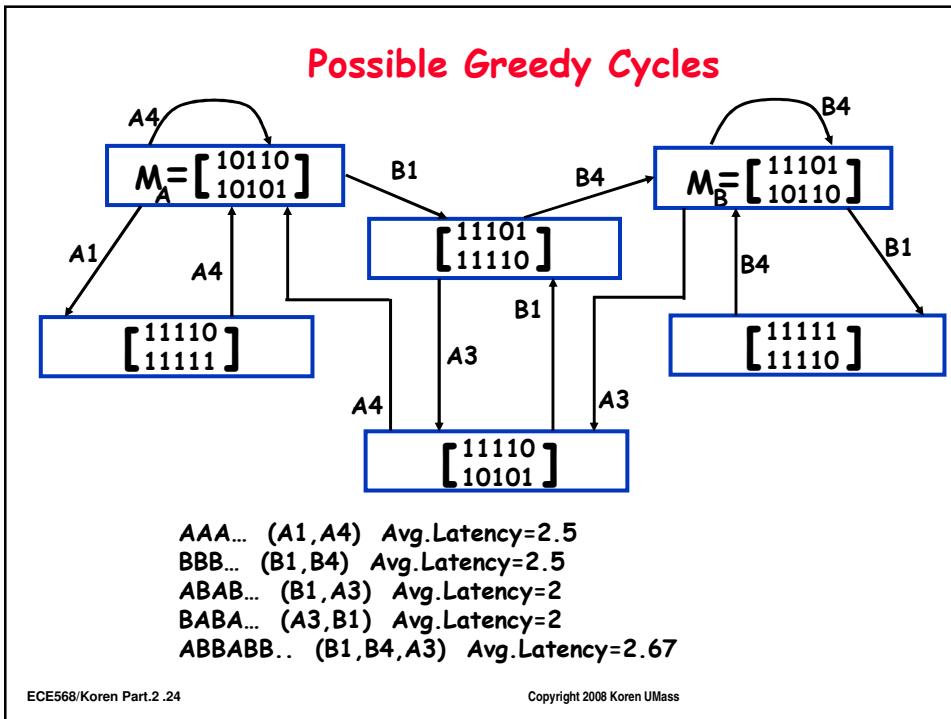
ECE568/Koren Part.2 .22

Copyright 2008 Koren UMass

State Diagram



Possible Greedy Cycles



Inserting delays to improve throughput

	t0	t1	t2	t3
s0	X	X		
s1	X		X	
s2	X			X

PCV=1111 → throughput=1/4 but MAL=4>2

After delay insertion
PCV=100100
greedy cycle=(1,1,4)
throughput=?

Is the solution unique?

	t0	t1	t2	t3	t4	t5
s0	X			X		
s1	D	X			X	
s2	D	D	X			X

Is it always advisable to insert delays?

# Ops.	RT	M-RT	Pi
1	4 <	6	0.5
2	8 >	7	0.4
3	12 >	8	0.05
4	16 >	12	0.05

Pi=Prob { i consecutive identical instructions}

RT-Avg.Latency=

M-RT-Avg.Latency=