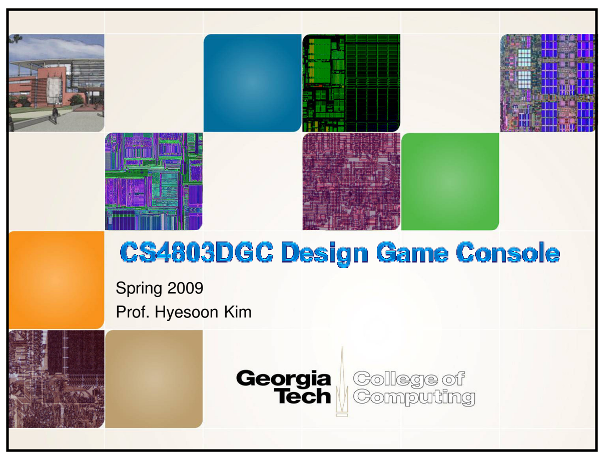
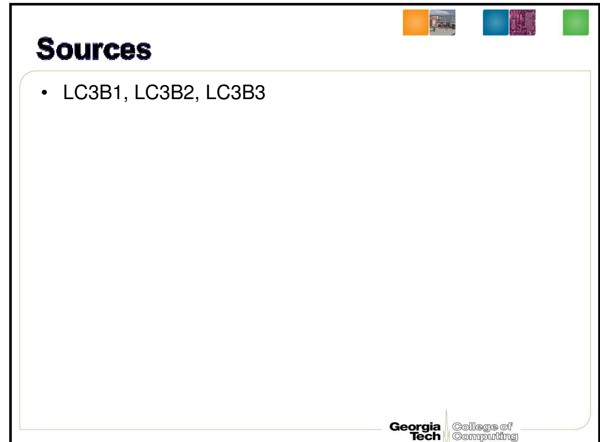
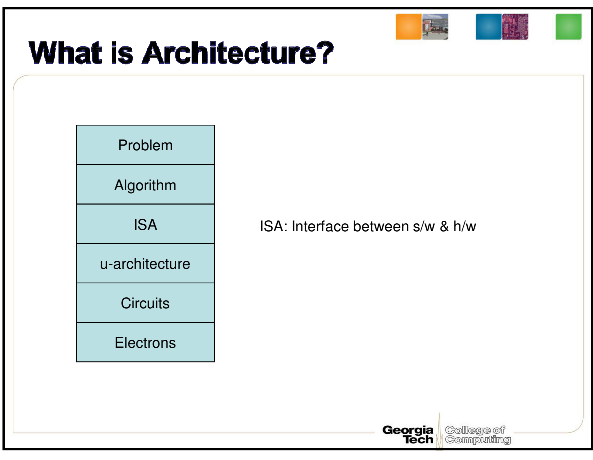
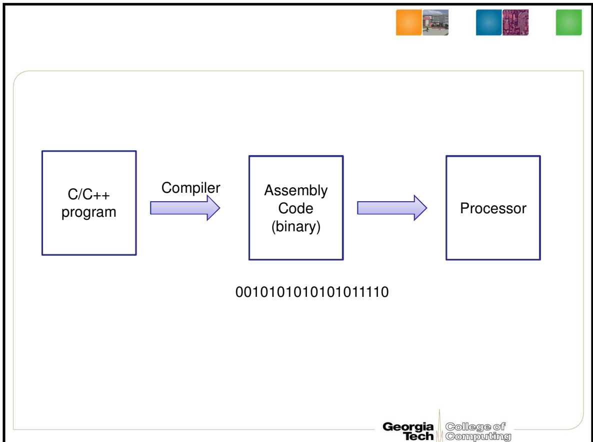

CS4803DGC Design Game Console
Spring 2009
Prof. Hyesoon Kim
Georgia Tech
College of Computing

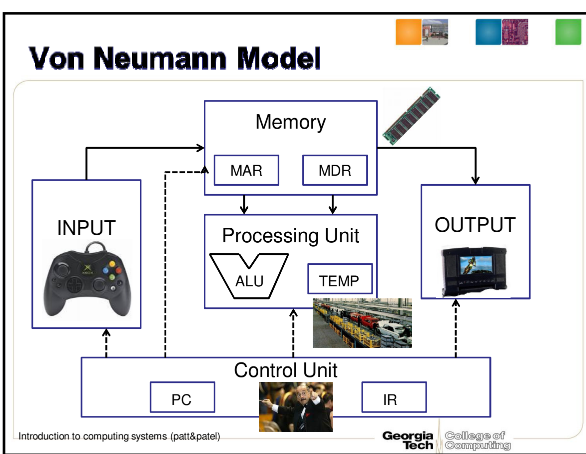

Sources
• LC3B1, LC3B2, LC3B3
Georgia Tech
College of Computing

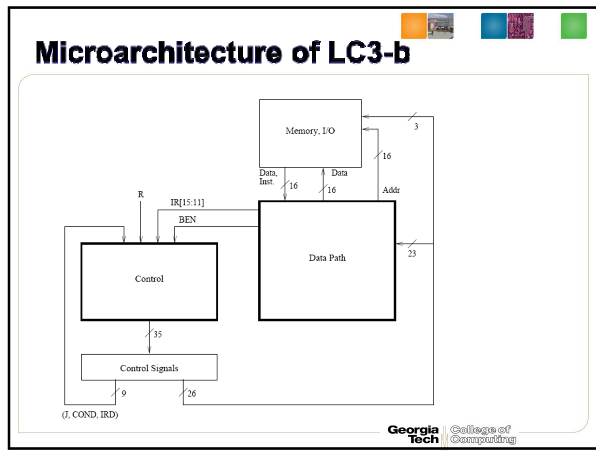

What is Architecture?
Problem
Algorithm
ISA
u-architecture
Circuits
Electrons
ISA: Interface between s/w & h/w
Georgia Tech
College of Computing

ISA (Instruction Set Architecture)

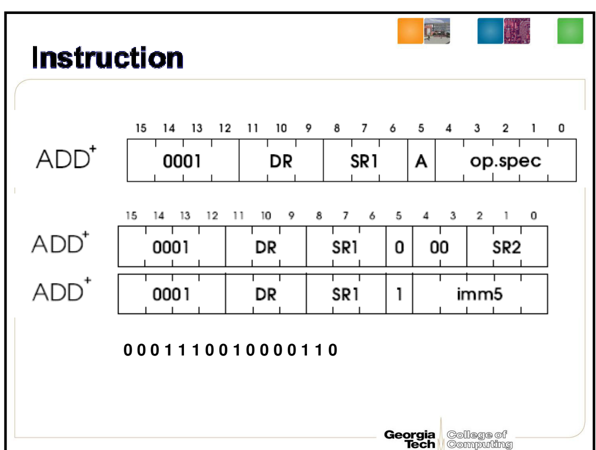
- Opcode
- Data type
- Addressing mode: register, immediate, displacement
  - E.g.) x86 SIB
- Memory addressing: unaligned, aligned



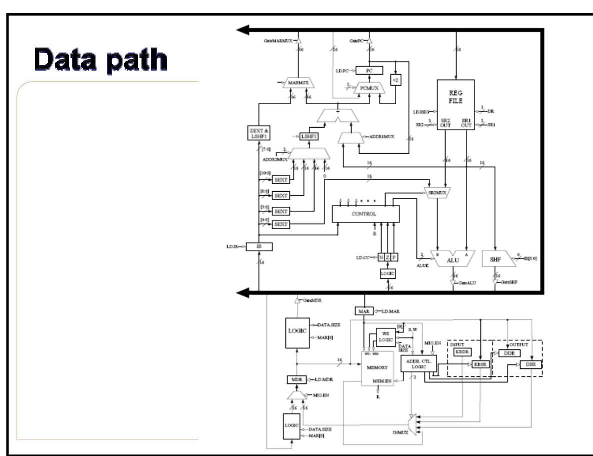




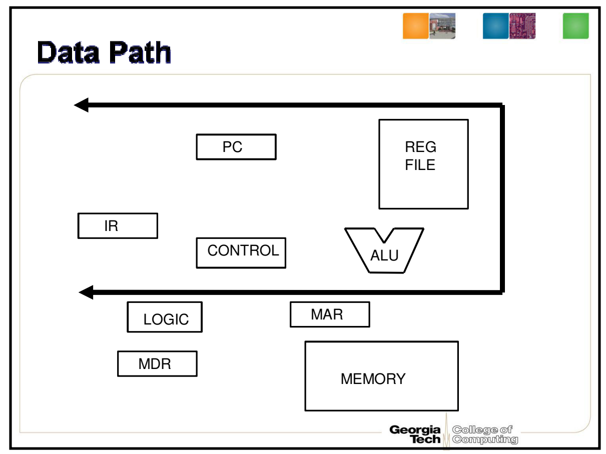
Microarchitecture of LC3-b
Memory, I/O
Data, Inst.
16
Data
16
16
3
Addr
R
IR[15:11]
BEN
Control
Data Path
23
35
Control Signals
9
26
(J, COND, IRD)
Georgia Tech
College of Computing



Instruction
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ADD+
0001
DR
SR1
A
op.spec
ADD+
0001
DR
SR1
0
00
SR2
ADD+
0001
DR
SR1
1
imm5
0001110010000110
Georgia Tech
College of Computing



Data path
REG FILE
CONTROL
ALU
MEMORY



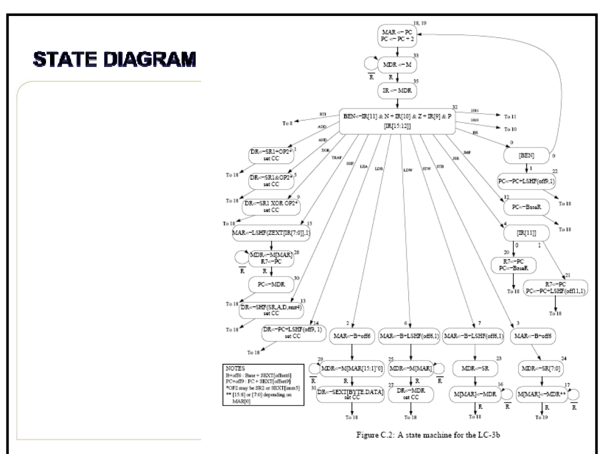
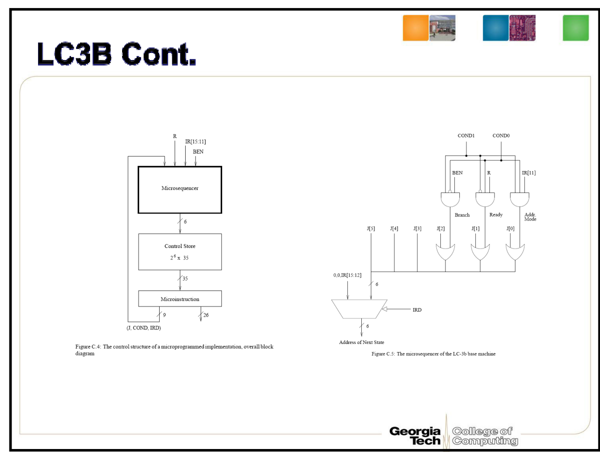


Figure C.2: A state machine for the LC-3b

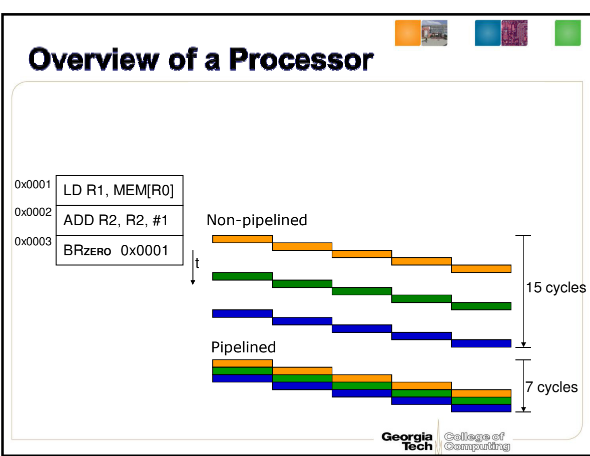


Homework #1

- Please read LC3b1, LC3b2, LC3b3 first.
- Be careful with Conditional Code

Georgia Tech College of Computing

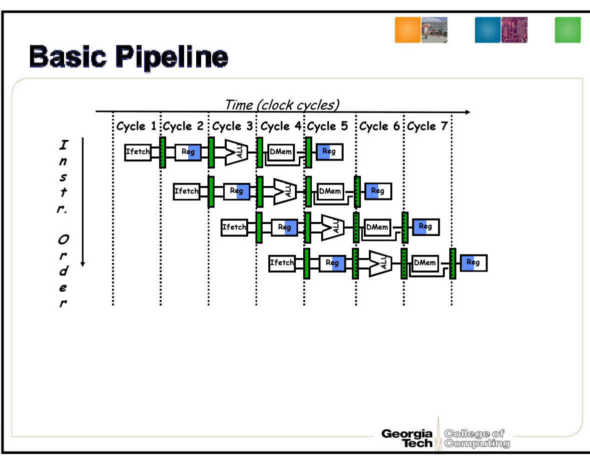
Overview of a Processor

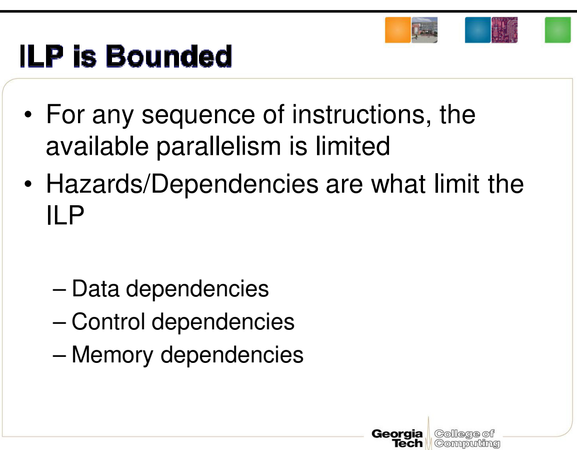


Simple 5-stage Pipeline

- IF: Instruction fetch cycle
- ID: Instruction decode/register fetch cycle
- EX: Execution/effective address cycle
- MEM: Memory access cycle
- WB: Write-back cycle

Georgia Tech College of Computing





Dependences/Dependencies

- Data Dependencies
  - RAW: Read-After-Write (True Dependence)
  - WAR: Anti-Depedence
  - WAW: Output Dependence
- Control Dependence
  - When following instructions depend on the outcome of a previous branch/jump

Ifetch Reg ALU DMem Reg

Georgia Tech College of Computing

Data Dependencies

- Register dependencies
  - RAW, WAR, WAW, based on register *number*
- Memory dependencies
  - Based on memory *address*
  - This is harder
    - Register names known at decode
    - Memory addresses not known until execute
    - Also called memory disambiguation

Georgia Tech College of Computing

Types of Data Dependencies

(Assume A comes before B in program order)

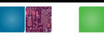
- RAW (Read-After-Write)
  - A writes to a location, B reads from the location, therefore B has a RAW dependency on A
  - Also called a "true dependency"

Georgia Tech College of Computing

Data Dep's (cont'd)

- WAR (Write-After-Read)
  - A reads from a location, B writes to the location, therefore B has a WAR dependency on A
  - If B executes before A has read its operand, then the operand will be lost
  - Also called an anti-dependence

Georgia Tech College of Computing

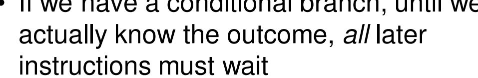


Data Dep's (cont'd)

- Write-After-Write
  - A writes to a location, B writes to the same location
  - If B writes first, then A writes, the location will end up with the wrong value
  - Also called an output-dependence

Georgia Tech College of Computing

Control Dependencies



- If we have a conditional branch, until we actually know the outcome, *all* later instructions must wait
  - That is, all instructions are control dependent on all earlier branches
  - This is true for unconditional branches as well (e.g., can't return from a function until we've loaded the return address)

Georgia Tech College of Computing

Memory Dependencies

- Basically similar to regular (register) data dependencies: RAW, WAR, WAW
- However, the exact location is not known until run-time :
  - A: STORE R1, 0[R2]
  - B: LOAD R5, 24[R8]
  - C: STORE R3, -8[R9]

  - RAW exists if (R2+0) == (R8+24)
  - WAR exists if (R8+24) == (R9 – 8)
  - WAW exists if (R2+0) == (R9 – 8)

Georgia Tech College of Computing

Identifying Dependences

I1: R2 = 17
I2: R1 = 49
I3: R3 = -8
I4: R5 = LOAD 0[R3]
I5: R4 = R1 + R2
I6: R7 = R4 – R3
I7: R6 = R4 * R5
I8: R5 = R6 + 10

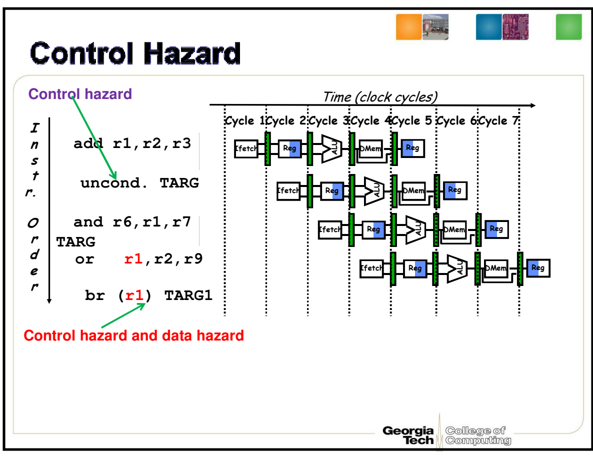
List RAW, WAR, WAW

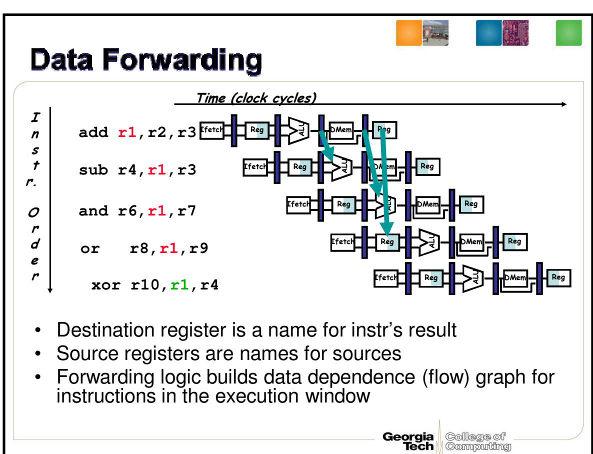
Georgia Tech College of Computing

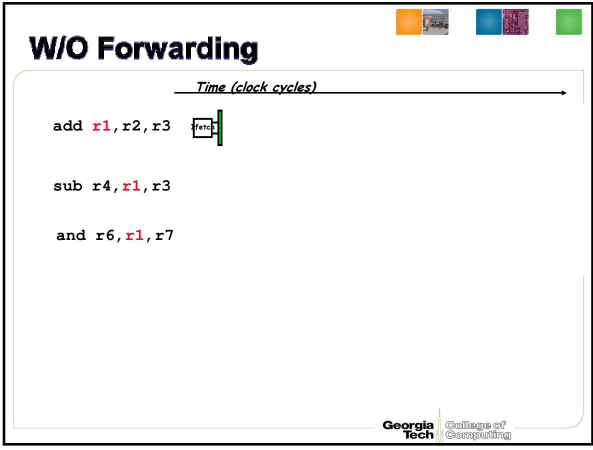
Hazards

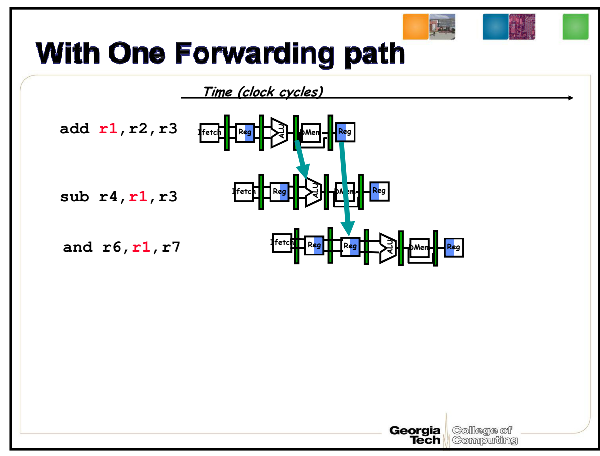
- When two instructions that have one or more dependences between them occur close enough that changing the instruction order will change the outcome of the program
- Not all dependencies lead to hazards!

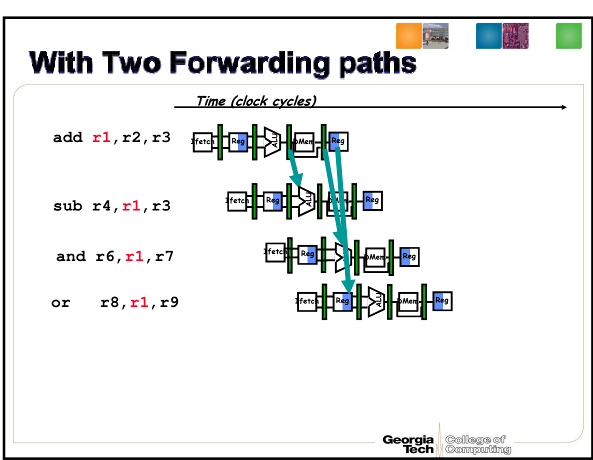
Georgia Tech College of Computing

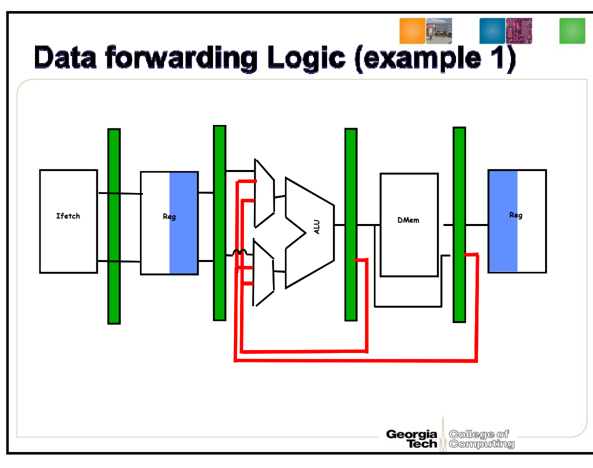

Control Hazard
Control hazard
Time (clock cycles)
Cycle 1 Cycle 2 Cycle 3 Cycle 4 Cycle 5 Cycle 6 Cycle 7
Instr. Order
add r1,r2,r3
uncond. TARG
and r6,r1,r7
TARG
or r1,r2,r9
br (r1) TARG1
Control hazard and data hazard
Georgia Tech College of Computing

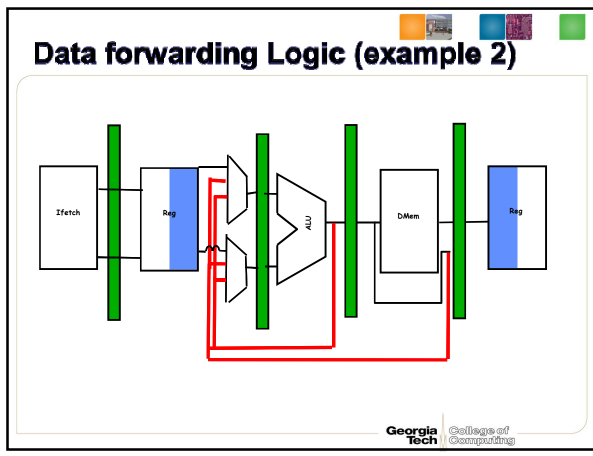

Data Forwarding
Time (clock cycles)
Instr. Order
add r1,r2,r3
sub r4,r1,r3
and r6,r1,r7
or r8,r1,r9
xor r10,r1,r4
• Destination register is a name for instr's result
• Source registers are names for sources
• Forwarding logic builds data dependence (flow) graph for instructions in the execution window
Georgia Tech College of Computing

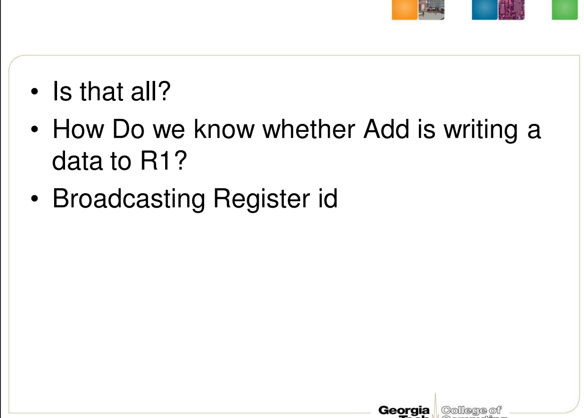

W/O Forwarding
Time (clock cycles)
add r1,r2,r3
sub r4,r1,r3
and r6,r1,r7
Georgia Tech College of Computing

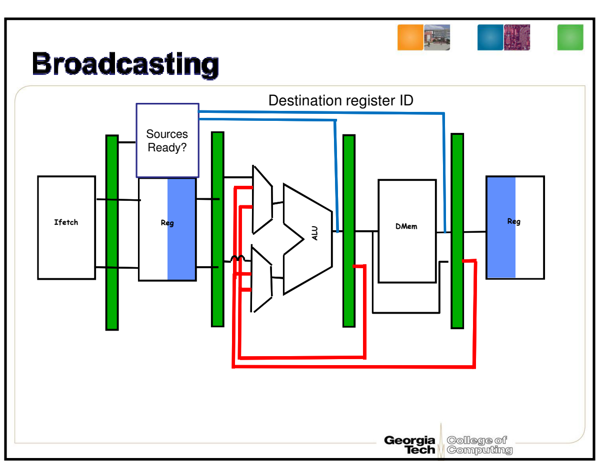

With One Forwarding path
Time (clock cycles)
add r1,r2,r3
sub r4,r1,r3
and r6,r1,r7
Georgia Tech
College of Computing

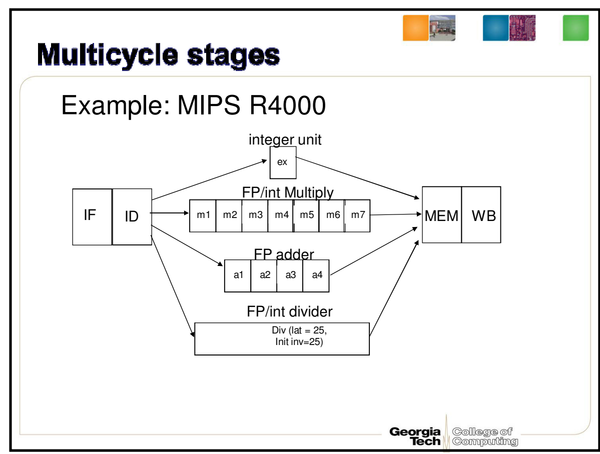

With Two Forwarding paths
Time (clock cycles)
add r1,r2,r3
sub r4,r1,r3
and r6,r1,r7
or r8,r1,r9
Georgia Tech
College of Computing

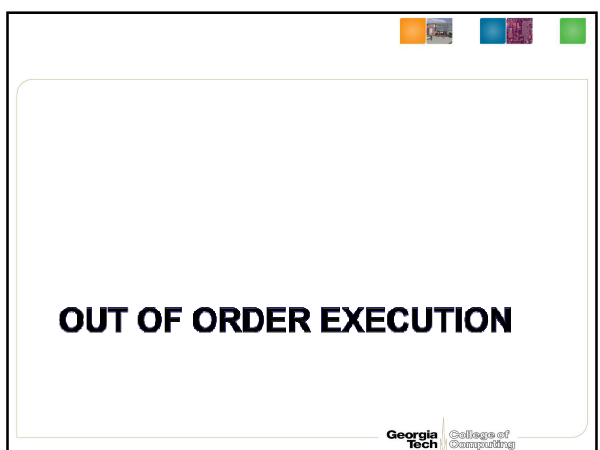

Data forwarding Logic (example 1)
Ifetch
Reg
ALU
DMem
Reg
Georgia Tech
College of Computing

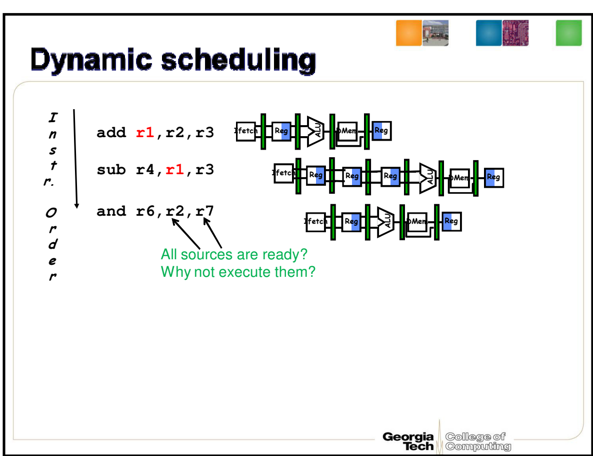

Data forwarding Logic (example 2)
Ifetch
Reg
ALU
DMem
Reg
Georgia Tech
College of Computing

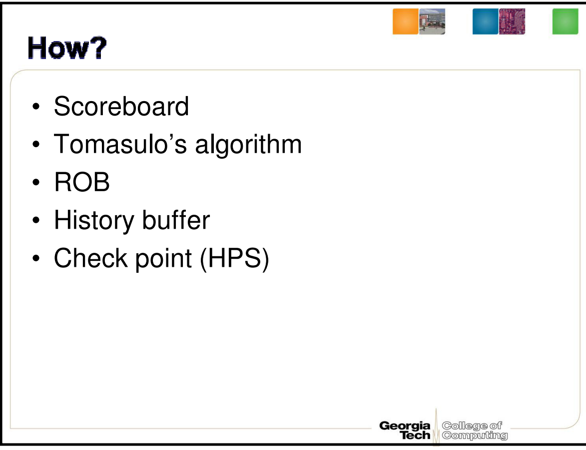

• Is that all?
• How Do we know whether Add is writing a data to R1?
• Broadcasting Register id
Georgia Tech
College of Computing

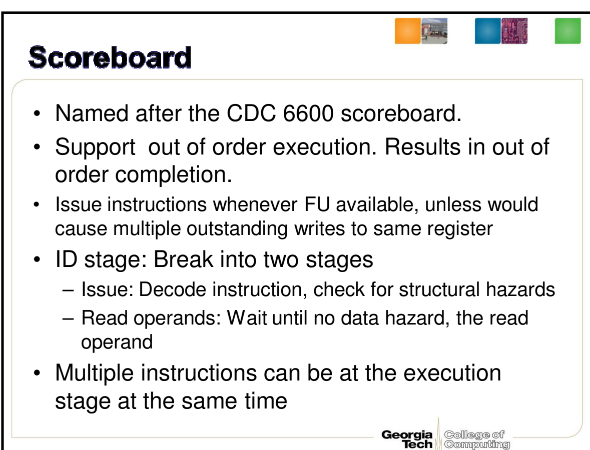

Broadcasting
Destination register ID
Sources Ready?
Ifetch
Reg
ALU
DMem
Reg
Georgia Tech
College of Computing

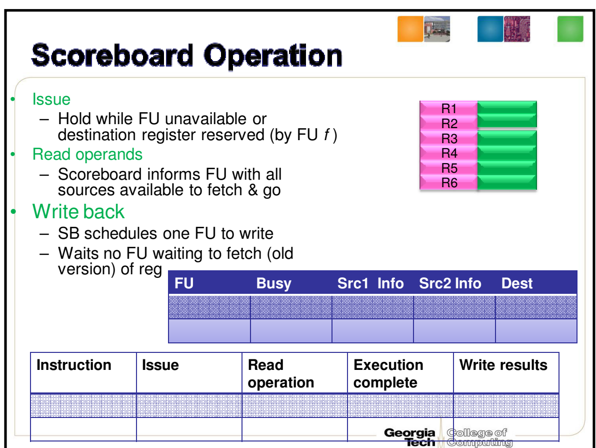

Multicycle stages
Example: MIPS R4000
integer unit
ex
FP/int Multiply
IF
ID
m1
m2
m3
m4
m5
m6
m7
MEM
WB
FP adder
a1
a2
a3
a4
FP/int divider
Div (lat = 25,
Init inv=25)
Georgia Tech
College of Computing

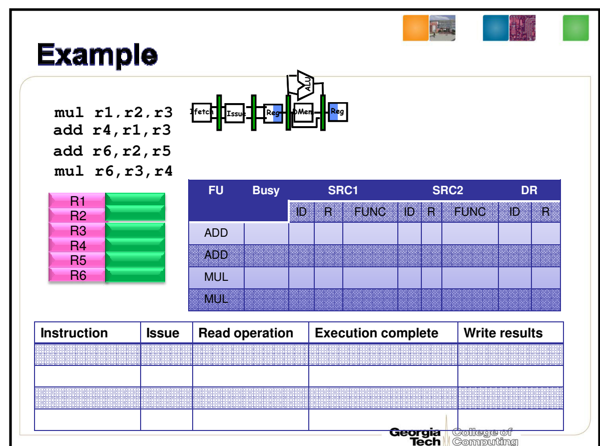

OUT OF ORDER EXECUTION
Georgia Tech
College of Computing

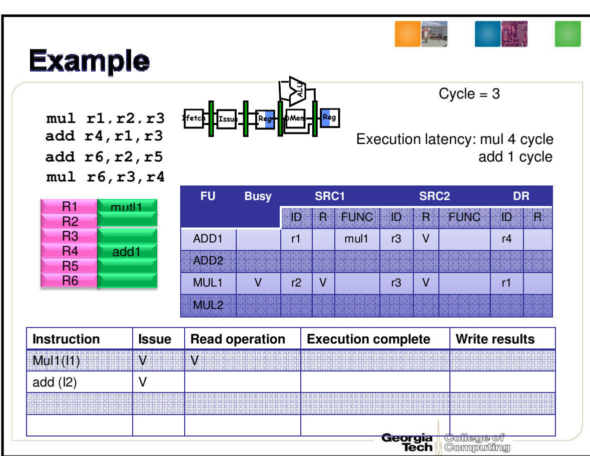

Dynamic scheduling
Instr. Order
add r1,r2,r3
sub r4,r1,r3
and r6,r2,r7
All sources are ready?
Why not execute them?
Georgia Tech
College of Computing

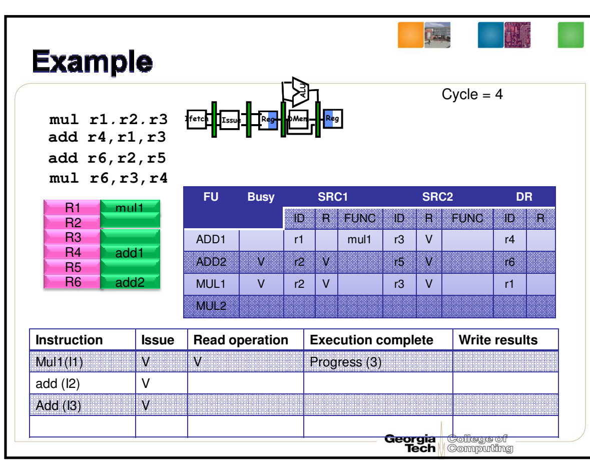

How?
Scoreboard
Tomasulo's algorithm
ROB
History buffer
Check point (HPS)
Georgia Tech College of Computing

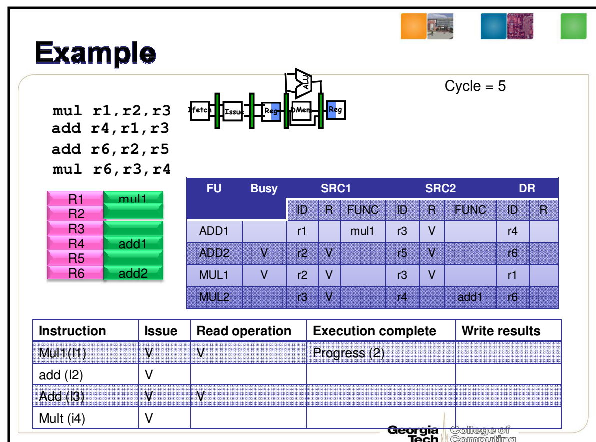

Scoreboard
Named after the CDC 6600 scoreboard.
Support out of order execution. Results in out of order completion.
Issue instructions whenever FU available, unless would cause multiple outstanding writes to same register
ID stage: Break into two stages
Issue: Decode instruction, check for structural hazards
Read operands: Wait until no data hazard, the read operand
Multiple instructions can be at the execution stage at the same time
Georgia Tech College of Computing

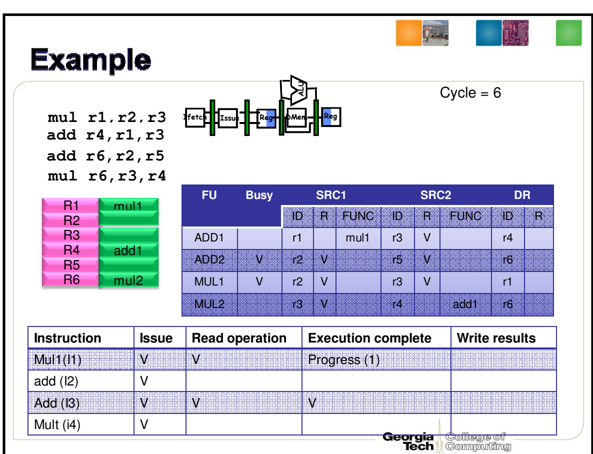

Scoreboard Operation
Issue
Hold while FU unavailable or destination register reserved (by FU f)
Read operands
Scoreboard informs FU with all sources available to fetch & go
Write back
SB schedules one FU to write
Waits no FU waiting to fetch (old version) of reg
R1
R2
R3
R4
R5
R6
FU
Busy
Src1 Info
Src2 Info
Dest
Instruction
Issue
Read operation
Execution complete
Write results
Georgia Tech College of Computing

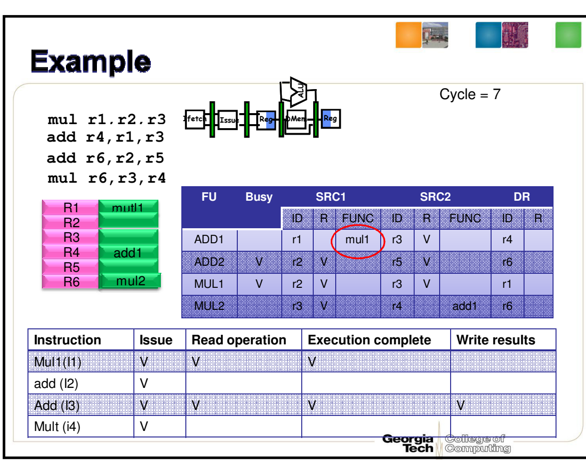

Example
mul r1,r2,r3
add r4,r1,r3
add r6,r2,r5
mul r6,r3,r4
R1
R2
R3
R4
R5
R6
FU
Busy
SRC1
SRC2
DR
ID
R
FUNC
ID
R
FUNC
ID
R
ADD
ADD
MUL
MUL
Instruction
Issue
Read operation
Execution complete
Write results
Georgia Tech
College of Computing

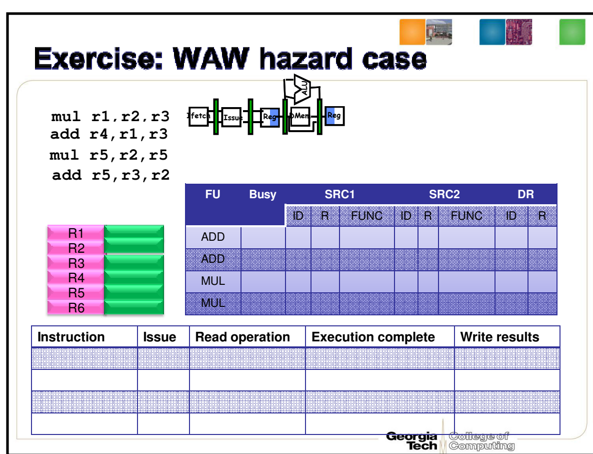

Example
Cycle = 3
mul r1,r2,r3
add r4,r1,r3
add r6,r2,r5
mul r6,r3,r4
Execution latency: mul 4 cycle
add 1 cycle
R1 mul1
R2
R3
R4 add1
R5
R6
FU
Busy
SRC1
SRC2
DR
ID
R
FUNC
ID
R
FUNC
ID
R
ADD1
r1
mul1
r3
V
r4
ADD2
MUL1
V
r2
V
r3
V
r1
MUL2
Instruction
Issue
Read operation
Execution complete
Write results
Mul1 (I1)
V
V
add (I2)
V
Georgia Tech
College of Computing

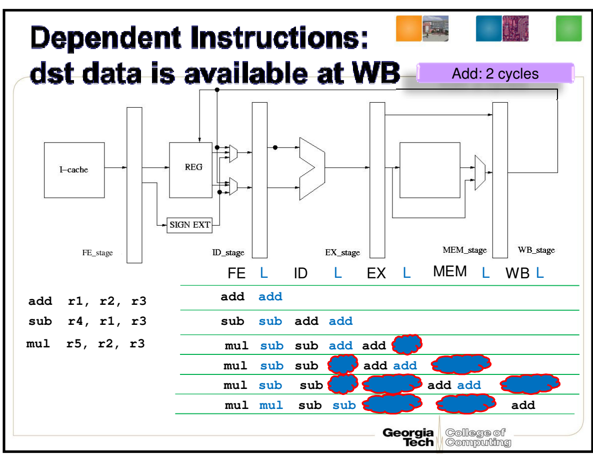

Example
Cycle = 4
mul r1,r2,r3
add r4,r1,r3
add r6,r2,r5
mul r6,r3,r4
R1 mul1
R2
R3
R4 add1
R5
R6 add2
FU
Busy
SRC1
SRC2
DR
ID
R
FUNC
ID
R
FUNC
ID
R
ADD1
r1
mul1
r3
V
r4
ADD2
V
r2
V
r5
V
r6
MUL1
V
r2
V
r3
V
r1
MUL2
Instruction
Issue
Read operation
Execution complete
Write results
Mul1 (I1)
V
V
Progress (3)
add (I2)
V
Add (I3)
V
Georgia Tech
College of Computing

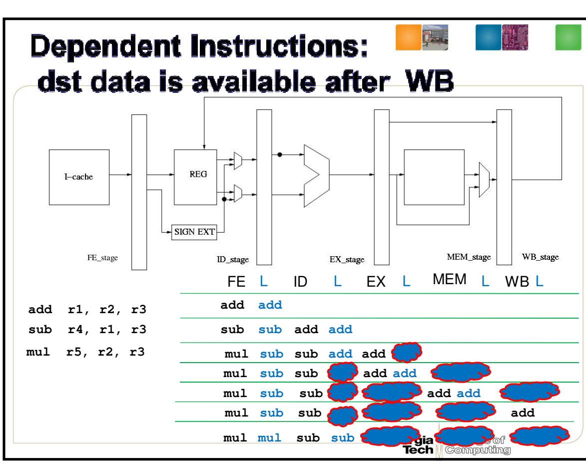

Example
Cycle = 5
mul r1,r2,r3
add r4,r1,r3
add r6,r2,r5
mul r6,r3,r4
R1 mul1
R2
R3
R4 add1
R5
R6 add2
FU | Busy | SRC1 ID | R | FUNC | SRC2 ID | R | FUNC | DR ID | R
ADD1 | | r1 | | mul1 | r3 | V | | r4 |
ADD2 | V | r2 | V | | r5 | V | | r6 |
MUL1 | V | r2 | V | | r3 | V | | r1 |
MUL2 | | r3 | V | | r4 | | add1 | r6 |
Instruction | Issue | Read operation | Execution complete | Write results
Mul1 (I1) | V | V | Progress (2) |
add (I2) | V | | |
Add (I3) | V | V | |
Mult (i4) | V | | |
Georgia Tech College of Computing

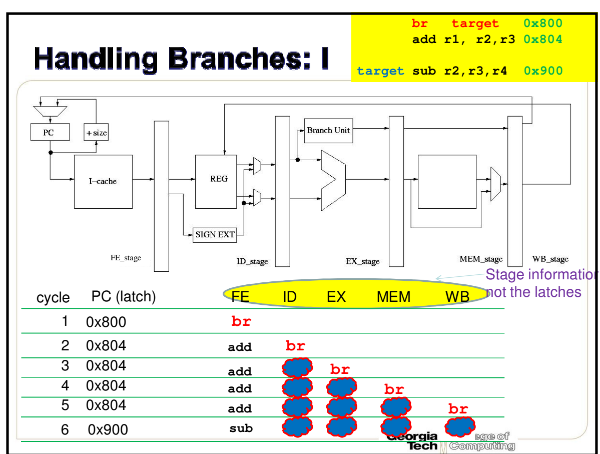

Example
Cycle = 6
mul r1,r2,r3
add r4,r1,r3
add r6,r2,r5
mul r6,r3,r4
R1 mul1
R2
R3
R4 add1
R5
R6 mul2
FU | Busy | SRC1 ID | R | FUNC | SRC2 ID | R | FUNC | DR ID | R
ADD1 | | r1 | | mul1 | r3 | V | | r4 |
ADD2 | V | r2 | V | | r5 | V | | r6 |
MUL1 | V | r2 | V | | r3 | V | | r1 |
MUL2 | | r3 | V | | r4 | | add1 | r6 |
Instruction | Issue | Read operation | Execution complete | Write results
Mul1 (I1) | V | V | Progress (1) |
add (I2) | V | | |
Add (I3) | V | V | V |
Mult (i4) | V | | |
Georgia Tech College of Computing

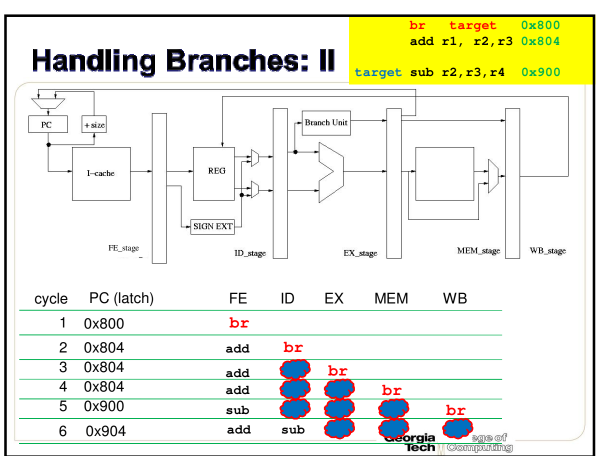

Example
Cycle = 7
mul r1,r2,r3
add r4,r1,r3
add r6,r2,r5
mul r6,r3,r4
R1 mul1
R2
R3
R4 add1
R5
R6 mul2
FU | Busy | SRC1 ID | R | FUNC | SRC2 ID | R | FUNC | DR ID | R
ADD1 | | r1 | | mul1 | r3 | V | | r4 |
ADD2 | V | r2 | V | | r5 | V | | r6 |
MUL1 | V | r2 | V | | r3 | V | | r1 |
MUL2 | | r3 | V | | r4 | | add1 | r6 |
Instruction | Issue | Read operation | Execution complete | Write results
Mul1 (I1) | V | V | V |
add (I2) | V | | |
Add (I3) | V | V | V | V
Mult (i4) | V | | |
Georgia Tech College of Computing

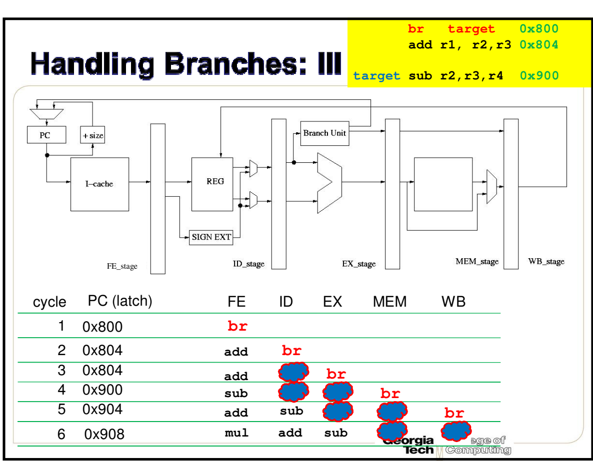

Exercise: WAW hazard case
mul r1,r2,r3
add r4,r1,r3
mul r5,r2,r5
add r5,r3,r2
FU Busy SRC1 SRC2 DR
ID R FUNC ID R FUNC ID R
ADD
ADD
MUL
MUL
R1
R2
R3
R4
R5
R6
Instruction Issue Read operation Execution complete Write results
Georgia Tech College of Computing

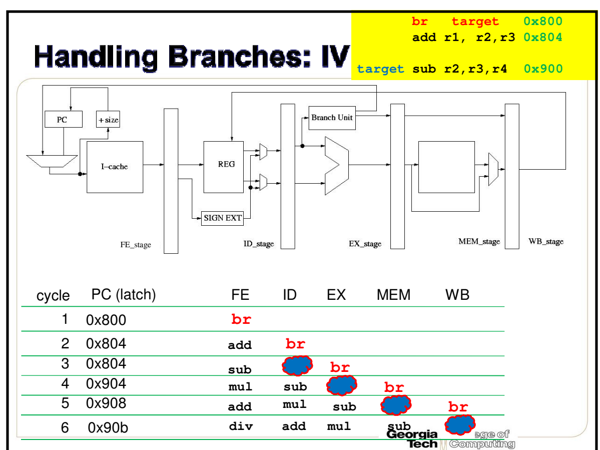

Dependent Instructions:
dst data is available at WB
Add: 2 cycles
I-cache
REG
SIGN EXT
FE_stage
ID_stage
EX_stage
MEM_stage
WB_stage
FE L ID L EX L MEM L WB L
add r1, r2, r3
sub r4, r1, r3
mul r5, r2, r3
add add
sub sub add add
mul sub sub add add
mul sub sub add add
mul sub sub add add
mul mul sub sub add
Georgia Tech College of Computing

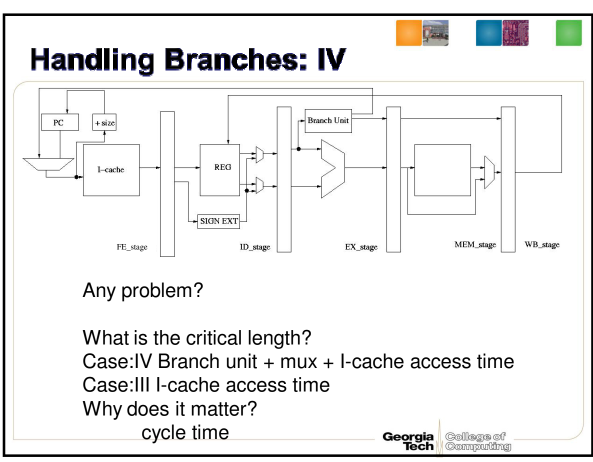

Dependent Instructions:
dst data is available after WB
I-cache
REG
SIGN EXT
FE_stage
ID_stage
EX_stage
MEM_stage
WB_stage
FE L ID L EX L MEM L WB L
add r1, r2, r3
sub r4, r1, r3
mul r5, r2, r3
add add
sub sub add add
mul sub sub add add
mul sub sub add add
mul sub sub add add
mul sub sub add
mul mul sub sub

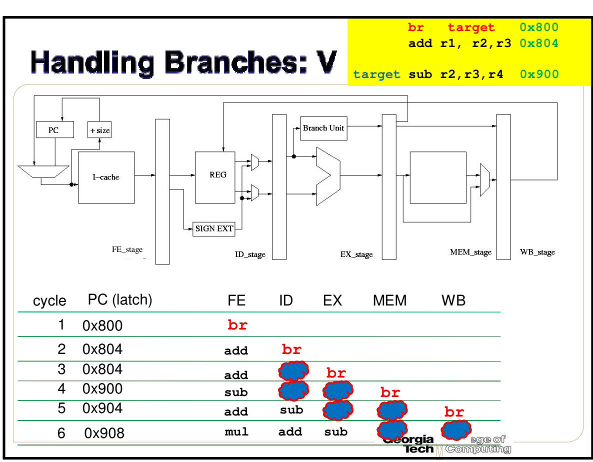

Handling Branches: I
br target 0x800
add r1, r2,r3 0x804
target sub r2,r3,r4 0x900
PC
+size
I-cache
REG
SIGN EXT
Branch Unit
FE_stage
ID_stage
EX_stage
MEM_stage
WB_stage
Stage information not the latches
cycle PC (latch) FE ID EX MEM WB
1 0x800 br
2 0x804 add br
3 0x804 add br
4 0x804 add br
5 0x804 add br
6 0x900 sub

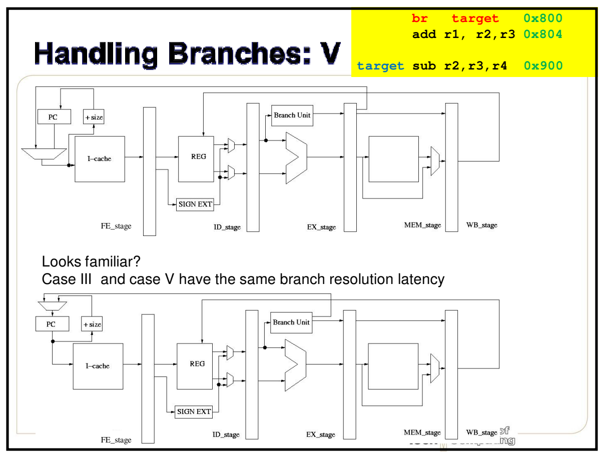

Handling Branches: II
br target 0x800
add r1, r2,r3 0x804
target sub r2,r3,r4 0x900
PC
+size
I-cache
REG
SIGN EXT
Branch Unit
FE_stage
ID_stage
EX_stage
MEM_stage
WB_stage
cycle PC (latch) FE ID EX MEM WB
1 0x800 br
2 0x804 add br
3 0x804 add br
4 0x804 add br
5 0x900 sub br
6 0x904 add sub

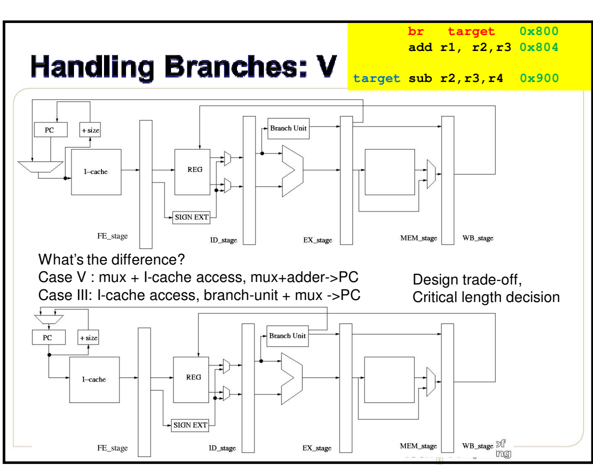

Handling Branches: III
br target 0x800
add r1, r2,r3 0x804
target sub r2,r3,r4 0x900
PC
+size
I-cache
REG
SIGN EXT
Branch Unit
FE_stage
ID_stage
EX_stage
MEM_stage
WB_stage
cycle PC (latch) FE ID EX MEM WB
1 0x800 br
2 0x804 add br
3 0x804 add br
4 0x900 sub br
5 0x904 add sub br
6 0x908 mul add sub

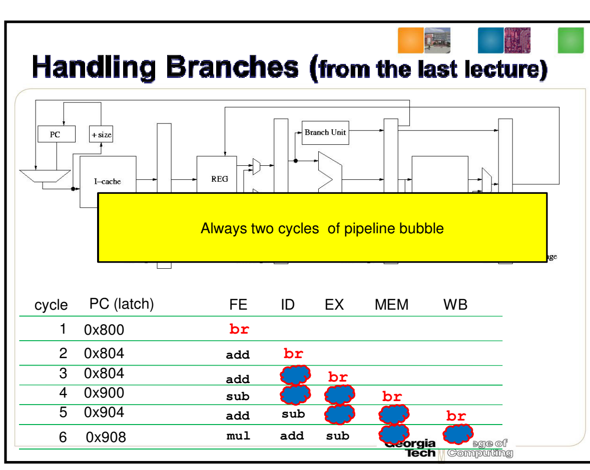

Handling Branches: IV
br target 0x800
add r1, r2,r3 0x804
target sub r2,r3,r4 0x900
PC
+ size
I-cache
REG
SIGN EXT
Branch Unit
FE_stage
ID_stage
EX_stage
MEM_stage
WB_stage
cycle PC (latch) FE ID EX MEM WB
1 0x800 br
2 0x804 add br
3 0x804 sub br
4 0x904 mul sub br
5 0x908 add mul sub br
6 0x90b div add mul sub
Georgia Tech


Handling Branches: IV
PC
+ size
I-cache
REG
SIGN EXT
Branch Unit
FE_stage
ID_stage
EX_stage
MEM_stage
WB_stage
Any problem?
What is the critical length?
Case:IV Branch unit + mux + I-cache access time
Case:III I-cache access time
Why does it matter?
cycle time
Georgia Tech
College of Computing

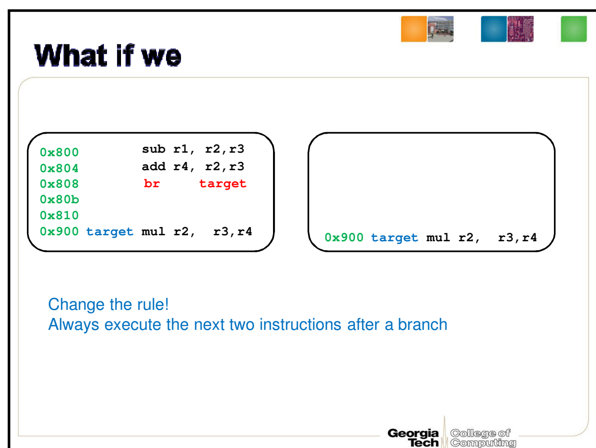

Handling Branches: V
br target 0x800
add r1, r2,r3 0x804
target sub r2,r3,r4 0x900
PC
+ size
I-cache
REG
SIGN EXT
Branch Unit
FE_stage
ID_stage
EX_stage
MEM_stage
WB_stage
cycle PC (latch) FE ID EX MEM WB
1 0x800 br
2 0x804 add br
3 0x804 add br
4 0x900 sub br
5 0x904 add sub br
6 0x908 mul add sub

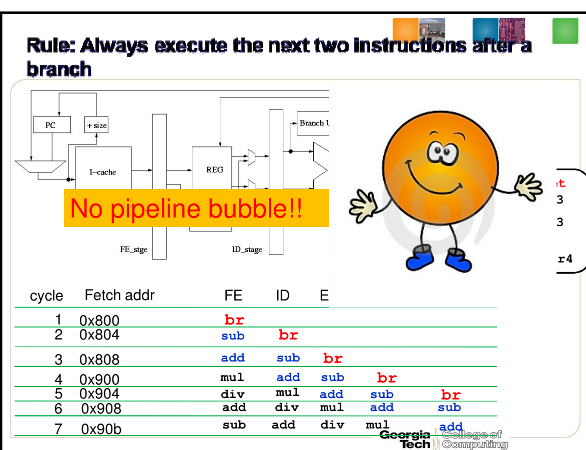


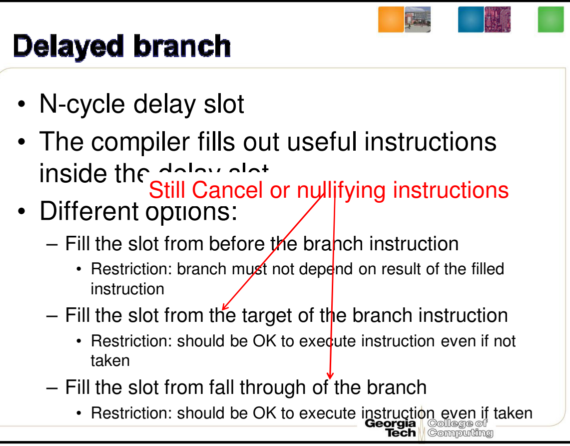




cycle	PC (latch)	FE	ID	EX	MEM	WB
1	0x800	br				
2	0x804	add	br			
3	0x804	add		br		
4	0x900	sub			br	
5	0x904	add	sub			br
6	0x908	mul	add	sub		


What if we
0x800 sub r1, r2,r3
0x804 add r4, r2,r3
0x808 br target
0x80b
0x810
0x900 target mul r2, r3,r4
0x900 target mul r2, r3,r4
Change the rule!
Always execute the next two instructions after a branch
Georgia Tech College of Computing


Rule: Always execute the next two instructions after a branch
No pipeline bubble!!
cycle Fetch addr FE ID E
1 0x800 br
2 0x804 sub br
3 0x808 add sub br
4 0x900 mul add sub br
5 0x904 div mul add sub br
6 0x908 add div mul add sub
7 0x90b sub add div mul add
Georgia Tech College of Computing


Delayed branch
• N-cycle delay slot
• The compiler fills out useful instructions inside the delay slot
• Different options:
– Fill the slot from before the branch instruction
• Restriction: branch must not depend on result of the filled instruction
– Fill the slot from the target of the branch instruction
• Restriction: should be OK to execute instruction even if not taken
– Fill the slot from fall through of the branch
• Restriction: should be OK to execute instruction even if taken
Still Cancel or nullifying instructions
Georgia Tech College of Computing

Remark

- Many DSP architecture, older RISC, MIPS, PA-RISC, SPARC.
- Delayed branches are architecturally ~~invisible~~ visible
  - Advantage:
    - better performance
  - Disadvantage:
    - what if implementation changes?
    - Deeper pipeline-> more branch delays?
- Interrupt/exceptions?
  - Where to go back?
- Combining with a branch predictor?

Georgia Tech College of Computing