Advanced Computer Architecture
Lab 1: MIPS-R10000

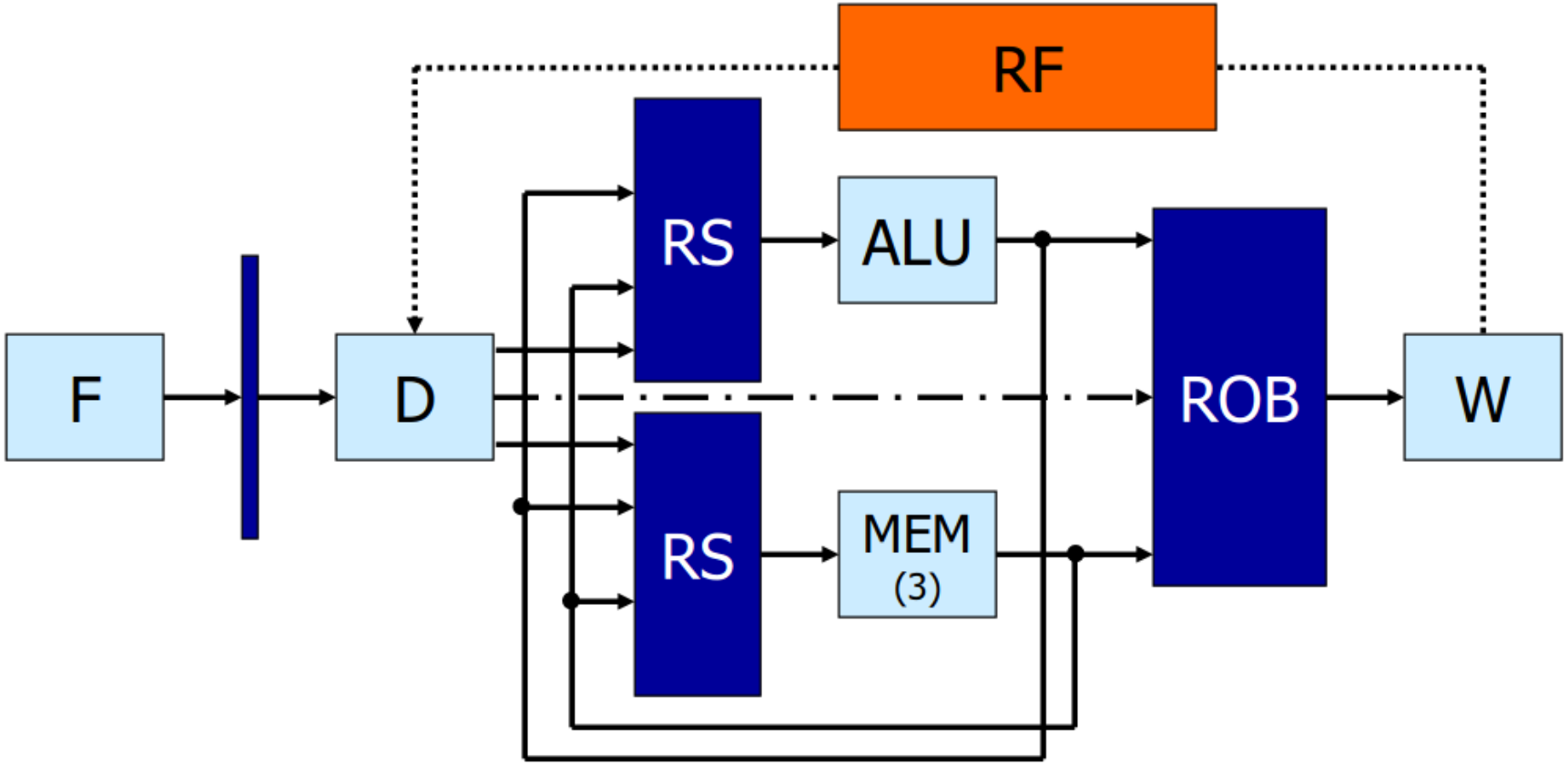
LAP

Outline

- MIPS R10000 is a superscalar processor from 1996
- Easy to understand:
  - Simple RISC ISA
  - Relatively old processor (many of the presented techniques have not changed)
- Great start before doing the homework!
  - Impossible to do the homework without understanding this lab very well
- Great opportunity to check your knowledge!
  - (Ask the TAs as many questions as you want)

Outline

- This processor view is great and high level but how things work in details?



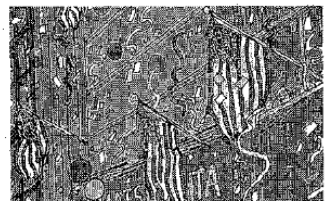
This lab

- First: Read the paper if not already done

THE MIPS R10000 SUPERSCALAR MICROPROCESSOR

Kenneth C. Yeager

Silicon Graphics, Inc.



The Mips R10000 is a dynamic, superscalar microprocessor that implements the 64-bit Mips 4 instruction set architecture. It fetches and decodes four instructions per cycle and dynamically issues them to five fully-pipelined, low-latency execution units. Instructions can be fetched and executed speculatively beyond branches. Instructions graduate in order upon completion. Although execution is out of order, the processor still provides sequential memory

parallel while the processor executes other instructions. This type of cache design is called "nonblocking," because cache refills do not block subsequent accesses to other cache lines.

Processors rely on compiler support to optimize instruction sequencing. This technique is especially effective for data arrays, such as those used in many floating-point applications. For these arrays, a sophisticated compiler can optimize performance for a spe-

This lab

- Answer the questions
- Ask the TAs whenever you have any doubt on the paper
- We are here to help!

1 Pipeline and Register Renaming

1. Explain why in the 3rd stage of the pipeline (see Figure 2) the *register file* is read in the 2nd half of each cycle. What are the advantages and disadvantages of such an implementation?

2. Figure 5 describes register renaming. Why are the destination and source registers represented with 5 bits in the original instruction, and with 6 bits after the renaming?

3. Specify the function of the *queues*, *active lists*, and *map tables* in the R10000. Relate each of these to the structures described in the course (*reservation stations*, *ROB*, ...).

4. Describe the role of the components *Rdy*, *OpA*, *OpB*, *OpC*, *Dest*, *Old Dest*

This lab

- Fill the simulation spreadsheet
- Good starting point before implementing the simulator (HW1)
- You cannot implement the simulator without understanding this lab very well
- Use the TAs
- Simulation details on the pdf on Moodle

Program

```
LDC1 $F0, #0000($I1)
LDC1 $F1, #0000($I1)
LDC1 $F2, #0000($I1)
LDC1 $F3, #0000($I1)
MUL.S $F4, $F0, $F3
MUL.S $F5, $F1, $F2
SUB.S $F4, $F4, $F5
ADD.S $F5,$F0, $F1
MUL.S $F9, $F5, $F5
MUL.S $F10, $F4, $F0
SUB.S $F9, $F9, $F10
SQRT.S $F9, $F9
SDC1 F4, #0004($I1)
SDC1 F5, #0005($I1)
SDC1 F9, #0006($I1)
SDC1 $F10, #0007($I1)
```

Simulation

End of Cycle #0

Active List 1			
Adress	Old Dest	Log Dest	Done
x00			
x01			
x02			
x03			
x04			
x05			
x06			
x07			

Active List 2			
Adress	Old Dest	Log Dest	Done
x08			
x09			
x0A			
x0B			
x0C			
x0D			
x0E			
x0F			

Active List 3			
Adress	Old Dest	Log Dest	Done
x10			
x11			
x12			
x13			
x14			
x15			
x16			
x17			

Active List 4			
Adress	Old Dest	Log Dest	Done
x18			
x19			
x1A			
x1B			
x1C			
x1D			
x1E			
x1F			

FP Register Map Table	
Logical	Map
x00	x00
x01	x01
x02	x02
x03	x03
x04	x04
x05	x05
x06	x06
x07	x07
x08	x08
x09	x09
x0A	x0A
x0B	x0B
x0C	x0C
x0D	x0D
x0E	x0E
x0F	x0F
x10	x10
x11	x11
x12	x12
x13	x13
x14	x14
x15	x15
x16	x16
x17	x17
x18	x18
x19	x19
x1A	x1A
x1B	x1B
x1C	x1C
x1D	x1D
x1E	x1E
x1F	x1F

Busy Bit Table			
x00	0	x20	0
x01	0	x21	0
x02	0	x22	0
x03	0	x23	0
x04	0	x24	0
x05	0	x25	0
x06	0	x26	0
x07	0	x27	0
x08	0	x28	0
x09	0	x29	0
x0A	0	x2A	0
x0B	0	x2B	0
x0C	0	x2C	0
x0D	0	x2D	0
x0E	0	x2E	0
x0F	0	x2F	0
x10	0	x30	0
x11	0	x31	0
x12	0	x32	0
x13	0	x33	0
x14	0	x34	0
x15	0	x35	0
x16	0	x36	0
x17	0	x37	0
x18	0	x38	0
x19	0	x39	0
x1A	0	x3A	0
x1B	0	x3B	0
x1C	0	x3C	0
x1D	0	x3D	0
x1E	0	x3E	0
x1F	0	x3F	0

Address Queue			
Tag	Rdy	OpA	Dest

Pipeline	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5	Stage 6	Stage 7
Instructions Positions	1 2 3 4						

FP Queue					
Tag	Rdy	OpA	OpB	OpC	Dest

Free List								
FIFO1	x20	x24	x28	x2C	x30	x34	x38	x3C
FIFO2	x21	x25	x29	x2D	x31	x35	x39	x3D
[illegible]	[illegible]	x26	x2A	x2E	x32	x36	x3A	x3E
FIFO4	x23	x27	x2B	x2F	x33	x37	x3B	x3F

FIFO Head

FIFO Tail

Solution to Lab questions

Pipelining and Renaming: Question 1

- Why is the register file read in the 2nd half of each cycle?

- We can write in the first half of the cycle and access the results in the same cycle
- (drawback: register file has to be fast)

Pipelining and Renaming: Question 2

- Why are the destination and source registers represented with 5 bits in original instructions and 6 bits after renaming?

- 32 logical registers, 64 physical registers.
- More parallelism available than what is exposed to programmer

Pipelining and Renaming: Question 3

- Meaning of queues, active lists, map tables

- Address Queue = Load Store Queue
- FP Queue = Reservation Station
- Active list = Reorder Buffer
- Register Map Table = Mapping Table

Pipelining and Renaming: Question 4

- Describe Rdy, OpA, OpB, OpC, Dest, Old Dest, Log Dest, Tag, D

- Rdy: Is the operand ready
- OpA, OpB, OpC: Physical registers of operands A, B, C
- Dest: Physical destination register
- Old Dest: Old physical register associated with logical
- Log Dest: Logical register in the program
- Tag: Addr within active list
- D: Has the instruction been executed yet?

Pipelining and Renaming: Question 5

- How many memory bits for FP queue, active list, map table, free register list, busy bit table?

- FP Queue:
  - Tag = 5 bits, Rdy= 3 bits,
  - OpA = OpB = OpC = Dest = 6 bits
  - Func = 10 bits
  - Unit = 4 bits
  - Br = 4 bits
  - 50 bits / line, 16 entires in the FP queue, 16 * 50 = 800 bits.

Pipelining and Renaming: Question 5

- How many memory bits for FP queue, active list, map table, free register list, busy bit table?

- Active List:
  - OldDest = 6 bits
  - Log Dest = 5 bits
  - Done = 1 bit
  - Exc = 6 bits
  - 18 bits / entry, 32 entries in the active list, 18 * 32 = 576 bits.

Pipelining and Renaming: Question 5

- How many memory bits for FP queue, active list, map table, free register list, busy bit table?

- Register Map Table:
  - 32 entries each 6 bits = 192 bits
- Free Register List:
  - 32 entries each 6 bits = 192 bits
- Busy Bit table:
  - 64 physical registers = 64 bits

Pipelining and Renaming: Question 6

- How many read / write ports in the busy bit / FP map table?
- Busy Bit Table read ports:
  - Each instructions has up to 3 operands (opA, opB, opC)
  - 4 instructions decoded per cycle
  - 3 additional ports for special instructions
  - = 15 read ports
- Map Table:
  - 4 instructions decoded per cycle, 3 operands each, a destination register each
  - Read physical registers: 4*4=16
  - Write new renaming mappings: 4*1=4

Pipelining and Renaming: Question 7

- How dependencies are detected and modeled?

- WAW, WAR: Disappear because of register renaming (become writes to different registers)
- RAW: busy bit table tracks operand dependencies in the reservation stations

Exception Handling: Question 1

- How R10000 handles exceptions?

- Discards all instructions after the one triggering exception in the ROB (program order)
- Unmaps 4 instructions per cycle
  - Need to check the oldDest of the Map table each time to restore mappings
- Waits for instructions before the one triggering exception to be done before jumping to exception handler

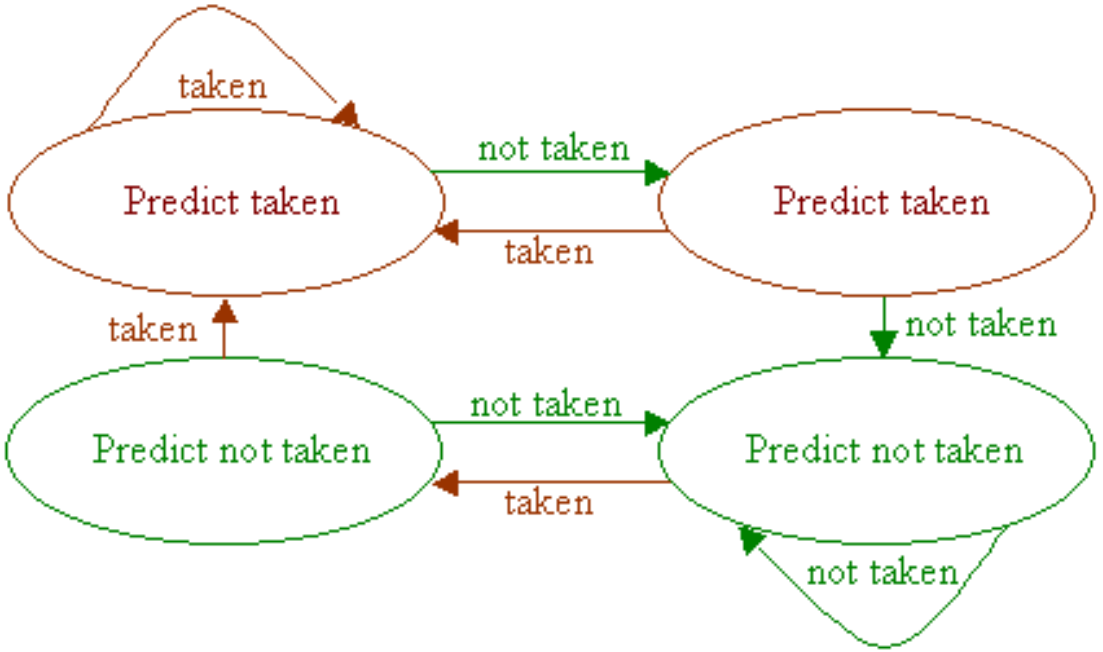
Exception Handling: Question 2

- What happens if instruction 6 generates an overflow?

- Going from the newest instruction in the active list to oldest, 4 instructions per cycle:
  - Read oldDest in the active list and updates the register map table
  - Removes corresponding entries from reservation stations

Branch Prediction: Question 1

- What algorithm R10000 uses for branch prediction?



Branch Prediction: Question 2

- What if a prediction is not correct?
- If the prediction is not correct, processor needs to restore state as at the time of the branch. For R10000:
  - R10000 can predict at most 4 branches at a time
  - Each time a prediction occurs, store the processor state in the Branch Stack
  - Branch stack contains:
    - the alternative address associated to the branch (the correct one)
    - Copies of register map tables
  - Processor copies content of the Branch Stack to corresponding structures
  - Processor discards any instruction fetched along mispredicted path using the branch mask
  - This operation takes a cycle, and no instruction can be decoded in the meantime

Branch Prediction: Question 3

- Why is there a NOP after the branch in the given program?

- In the MIPS ISA, any instruction right after a branch is always executed. The NOP is necessary for program correctness when no instruction independent from the branch can be scheduled at this slot.