Superscalar Complexity; Road to Multithreading
View of Superscalar Execution
Generic Superscalar Processor Models

Issue queue based

Fetch → Rename → Wakeup select → Regfile → bypass → FU → D-cache → commit
schedule
execute

Reservation based

Fetch → Rename → ROB → Wakeup select → Reg → bypass → FU → D-cache → commit
schedule
execute
Register Renaming

Instruction Fetch

Decode/Rename

Execute

Memory

Writeback

Instruction

Decode Logic

ADD R1, R2, R3

Rename Logic

ADD H7, H8, H9

Register Read

H7 = result

Register Write

H7 -> R1

Register Alias Table

R1 -> H7

R2 -> H8

R3 -> H9

Instruction Fetch

Decode/Rename

Execute

Memory

Writeback
Register Renaming

- Alpha 21264+, MIPS R10K+, Pentium 4 use explicit register renaming
  - Registers are not read until instruction dispatches (begins execution)
  - Register renaming ensures no conflicts

```plaintext
DIV R5, R4, R2
ADD R7, R5, R1
SUB R5, R3, R2
LW R7, 1000(R5)
```

<table>
<thead>
<tr>
<th>RAT</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>PR23</td>
</tr>
<tr>
<td>R2</td>
<td>PR2</td>
</tr>
<tr>
<td>R3</td>
<td>PR17</td>
</tr>
<tr>
<td>R4</td>
<td>PR45</td>
</tr>
<tr>
<td>R5</td>
<td>PR13</td>
</tr>
<tr>
<td>R6</td>
<td>PR20</td>
</tr>
<tr>
<td>R7</td>
<td>PR30</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Register Renaming

- Assume that PR37 is free and allocated to DIV’s destination register R5
Register Renaming

- Assume that \textcolor{red}{PR37} is free and allocated to DIV’s destination register R5

```
DIV R5, R4, R2  DIV PR37, PR45, PR2
ADD R7, R5, R1
SUB R5, R3, R2
LW R7, 1000(R5)
```
Assume that PR4 is free and allocated to ADD’s destination register R7
Register Renaming

DIV R5, R4, R2
ADD R7, R5, R1
SUB R5, R3, R2
LW  R7, 1000(R5)

DIV PR37, PR45, PR2
ADD PR4, PR37, PR23
SUB PR42, PR17, PR2

DIV R5, R4, R2
ADD R7, R5, R1
SUB R5, R3, R2
LW  R7, 1000(R5)

<table>
<thead>
<tr>
<th>RAT</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>PR23</td>
</tr>
<tr>
<td>R2</td>
<td>PR2</td>
</tr>
<tr>
<td>R3</td>
<td>PR17</td>
</tr>
<tr>
<td>R4</td>
<td>PR45</td>
</tr>
<tr>
<td>R5</td>
<td>PR37</td>
</tr>
<tr>
<td>R6</td>
<td>PR20</td>
</tr>
<tr>
<td>R7</td>
<td>PR4</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
## Register Renaming

<table>
<thead>
<tr>
<th>RAT</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>PR23</td>
</tr>
<tr>
<td>R2</td>
<td>PR2</td>
</tr>
<tr>
<td>R3</td>
<td>PR17</td>
</tr>
<tr>
<td>R4</td>
<td>PR45</td>
</tr>
<tr>
<td>R5</td>
<td>PR42</td>
</tr>
<tr>
<td>R6</td>
<td>PR20</td>
</tr>
<tr>
<td>R7</td>
<td>PR4</td>
</tr>
<tr>
<td>...</td>
<td>PR19</td>
</tr>
</tbody>
</table>

- `DIV R5, R4, R2`  
  `DIV PR37, PR45, PR2`
- `ADD R7, R5, R1`  
  `ADD PR4, PR37, PR23`
- `SUB R5, R3, R2`  
  `SUB PR42, PR17, PR2`
- `LW R7, 1000 (R5)`  
  `LW PR19, 1000 (PR42)`
MIPS R10000, Recycling Physical Registers

I1: DIV R5, R4, R2
I2: ADD R7, R5, R1
I3: SUB R5, R3, R2
I4: LW R7, 1000(R5)

Register Map

<table>
<thead>
<tr>
<th>Register</th>
<th>PR</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>PR23</td>
</tr>
<tr>
<td>R2</td>
<td>PR2</td>
</tr>
<tr>
<td>R3</td>
<td>PR17</td>
</tr>
<tr>
<td>R4</td>
<td>PR45</td>
</tr>
<tr>
<td>R5</td>
<td>PR13</td>
</tr>
<tr>
<td>R6</td>
<td>PR20</td>
</tr>
<tr>
<td>R7</td>
<td>PR30</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

Instruction Queue (Reservation Stations)

Active List

Register Free List

PR37, PR4, PR42, PR19, ...

Head
Tail
Commit
MIPS R10000, Recycling Physical Registers

I1: DIV R5, R4, R2
I2: ADD R7, R5, R1
I3: SUB R5, R3, R2
I4: LW R7, 1000(R5)

Register Map

<table>
<thead>
<tr>
<th>Register</th>
<th>Physical Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>PR23</td>
</tr>
<tr>
<td>R2</td>
<td>PR2</td>
</tr>
<tr>
<td>R3</td>
<td>PR17</td>
</tr>
<tr>
<td>R4</td>
<td>PR45</td>
</tr>
<tr>
<td>R5</td>
<td>PR13</td>
</tr>
<tr>
<td>R6</td>
<td>PR20</td>
</tr>
<tr>
<td>R7</td>
<td>PR30</td>
</tr>
</tbody>
</table>

Instruction Queue (Reservation Stations)

Active List

Register Free List

PR37, PR4, PR42, PR19, ...
MIPS R10000, Recycling Physical Registers

I1: DIV R5, R4, R2
I2: ADD R7, R5, R1
I3: SUB R5, R3, R2
I4: LW R7, 1000(R5)

Register Map

<table>
<thead>
<tr>
<th>Register</th>
<th>Physical Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>PR23</td>
</tr>
<tr>
<td>R2</td>
<td>PR2</td>
</tr>
<tr>
<td>R3</td>
<td>PR17</td>
</tr>
<tr>
<td>R4</td>
<td>PR45</td>
</tr>
<tr>
<td>R5</td>
<td>PR37</td>
</tr>
<tr>
<td>R6</td>
<td>PR20</td>
</tr>
<tr>
<td>R7</td>
<td>PR30</td>
</tr>
</tbody>
</table>

...  

Instruction Queue (Reservation Stations)

DIV PR37, PR45, PR2

Active List

Head

I1: PR13

Tail

Register Free List

PR4, PR42, PR19, ...

Commit
MIPS R10000, Recycling Physical Registers

I1: DIV R5, R4, R2
I2: ADD R7, R5, R1
I3: SUB R5, R3, R2
I4: LW R7, 1000(R5)

Register Map

<table>
<thead>
<tr>
<th>Register</th>
<th>PR</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>PR23</td>
</tr>
<tr>
<td>R2</td>
<td>PR2</td>
</tr>
<tr>
<td>R3</td>
<td>PR17</td>
</tr>
<tr>
<td>R4</td>
<td>PR45</td>
</tr>
<tr>
<td>R5</td>
<td>PR37</td>
</tr>
<tr>
<td>R6</td>
<td>PR20</td>
</tr>
<tr>
<td>R7</td>
<td>PR4</td>
</tr>
</tbody>
</table>

Instruction Queue (Reservation Stations)

ADD PR4, PR37, PR23
DIV 2, 45:37

Active List

Head
I1: PR13
I2: PR30
Tail

Register Free List

PR42, PR19, ...

Commit
MIPS R10000, Recycling Physical Registers

I1: DIV R5, R4, R2
I2: ADD R7, R5, R1
I3: SUB R5, R3, R2
I4: LW R7, 1000(R5)

Register Map

<table>
<thead>
<tr>
<th>Register</th>
<th>PR</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>PR23</td>
</tr>
<tr>
<td>R2</td>
<td>PR2</td>
</tr>
<tr>
<td>R3</td>
<td>PR17</td>
</tr>
<tr>
<td>R4</td>
<td>PR45</td>
</tr>
<tr>
<td>R5</td>
<td>PR42</td>
</tr>
<tr>
<td>R6</td>
<td>PR20</td>
</tr>
<tr>
<td>R7</td>
<td>PR4</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

Instruction Queue (Reservation Stations)

<table>
<thead>
<tr>
<th>Instruction</th>
<th>PR</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIV 2, 45:37</td>
<td></td>
</tr>
<tr>
<td>ADD 37, 23: 4</td>
<td></td>
</tr>
<tr>
<td>I1: PR13</td>
<td></td>
</tr>
<tr>
<td>I2: PR30</td>
<td></td>
</tr>
<tr>
<td>I3: PR37</td>
<td></td>
</tr>
</tbody>
</table>

Active List

Register Free List

<table>
<thead>
<tr>
<th>PR</th>
</tr>
</thead>
<tbody>
<tr>
<td>PR19, ...</td>
</tr>
</tbody>
</table>

Commit
MIPS R10000, Recycling Physical Registers

I1: DIV R5, R4, R2
I2: ADD R7, R5, R1
I3: SUB R5, R3, R2
I4: LW R7, 1000(R5)

Register Map

<table>
<thead>
<tr>
<th>Register</th>
<th>PR</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>PR23</td>
</tr>
<tr>
<td>R2</td>
<td>PR2</td>
</tr>
<tr>
<td>R3</td>
<td>PR17</td>
</tr>
<tr>
<td>R4</td>
<td>PR45</td>
</tr>
<tr>
<td>R5</td>
<td>PR42</td>
</tr>
<tr>
<td>R6</td>
<td>PR20</td>
</tr>
<tr>
<td>R7</td>
<td>PR19</td>
</tr>
</tbody>
</table>

Instruction Queue (Reservation Stations)

<table>
<thead>
<tr>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIV 2,45:37</td>
</tr>
<tr>
<td>ADD 37,23:4</td>
</tr>
<tr>
<td>SUB 17, 2:42</td>
</tr>
</tbody>
</table>

Active List

<table>
<thead>
<tr>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>I1: PR13</td>
</tr>
<tr>
<td>I2: PR30</td>
</tr>
<tr>
<td>I3: PR37</td>
</tr>
<tr>
<td>I4: PR4</td>
</tr>
</tbody>
</table>

Register Free List

| ... |

Commit
MIPS R10000, Recycling Physical Registers

I1: DIV R5, R4, R2
I2: ADD R7, R5, R1
I3: SUB R5, R3, R2
I4: LW R7, 1000(R5)

Register Map

<table>
<thead>
<tr>
<th>Register</th>
<th>PR</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>PR23</td>
</tr>
<tr>
<td>R2</td>
<td>PR2</td>
</tr>
<tr>
<td>R3</td>
<td>PR17</td>
</tr>
<tr>
<td>R4</td>
<td>PR45</td>
</tr>
<tr>
<td>R5</td>
<td>PR42</td>
</tr>
<tr>
<td>R6</td>
<td>PR20</td>
</tr>
<tr>
<td>R7</td>
<td>PR19</td>
</tr>
</tbody>
</table>

Instruction Queue (Reservation Stations)

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIV 2,45:37</td>
<td></td>
</tr>
<tr>
<td>ADD 37,23:4</td>
<td></td>
</tr>
<tr>
<td>SUB 17,2:42</td>
<td></td>
</tr>
<tr>
<td>LW 42:19</td>
<td></td>
</tr>
</tbody>
</table>

Active List

<table>
<thead>
<tr>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>I1: PR13</td>
</tr>
<tr>
<td>I2: PR30</td>
</tr>
<tr>
<td>I3: PR37</td>
</tr>
<tr>
<td>I4: PR4</td>
</tr>
</tbody>
</table>

Register Free List

...
MIPS R10000, Recycling Physical Registers

I1: \text{DIV} R5, R4, R2
I2: \text{ADD} R7, R5, R1
I3: \text{SUB} R5, R3, R2
I4: \text{LW} R7, 1000(R5)

\begin{center}
\begin{tabular}{|c|c|}
\hline
Register Map & Instruction Queue (Reservation Stations) \\
\hline
R1 & PR23 \text{ ADD 37,23: 4} \\
R2 & PR2 \\
R3 & PR17 \\
R4 & PR45 \\
R5 & PR42 \\
R6 & PR20 \\
R7 & PR19 \\
\hline
\end{tabular}
\end{center}

Active List

... 
I1: PR13 
I2: PR30 
I3: PR37 
I4: PR4

Register Free List

... 

Commit

I4, producing register 19, completes, broadcasts a completion signal to IQ
MIPS R10000, Recycling Physical Registers

I1: DIV R5, R4, R2
I2: ADD R7, R5, R1
I3: SUB R5, R3, R2
I4: LW R7, 1000(R5)

I1, producing register 37, completes, broadcasts a completion signal to IQ
MIPS R10000, Recycling Physical Registers

I1: DIV R5, R4, R2
I2: ADD R7, R5, R1
I3: SUB R5, R3, R2
I4: LW R7, 1000(R5)

Register Map

<table>
<thead>
<tr>
<th>Register</th>
<th>PR</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>PR23</td>
</tr>
<tr>
<td>R2</td>
<td>PR2</td>
</tr>
<tr>
<td>R3</td>
<td>PR17</td>
</tr>
<tr>
<td>R4</td>
<td>PR45</td>
</tr>
<tr>
<td>R5</td>
<td>PR42</td>
</tr>
<tr>
<td>R6</td>
<td>PR20</td>
</tr>
<tr>
<td>R7</td>
<td>PR19</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Instruction Queue (Reservation Stations)

<table>
<thead>
<tr>
<th>Instruction</th>
<th>PR</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD 37,23: 4</td>
<td>PR4</td>
</tr>
</tbody>
</table>

Active List

<table>
<thead>
<tr>
<th>Instruction</th>
<th>PR</th>
</tr>
</thead>
<tbody>
<tr>
<td>I1: PR13</td>
<td>PR30</td>
</tr>
<tr>
<td>I2: PR30</td>
<td>PR37</td>
</tr>
<tr>
<td>I3: PR37</td>
<td>PR4</td>
</tr>
</tbody>
</table>

Register Free List

... PR13

I2, producing register 4, completes, broadcasts a completion signal to IQ
I1, commits
Instr Scheduling: Wakeup & Select

- **Wakeup Logic**
  - To notify the resolution of data dependency of input operands
  - Wake up instructions with zero input dependency

- **Select Logic**
  - Choose and fire ready instructions
  - Deal with structure hazard

- **Wakeup-select is likely on the critical path**
  - Associative match
Scalar Scheduler (Issue Width = 1)

From Prof. G. Loh’s Slide
Superscalar Scheduler (Width = 4)

Tag Broadcast Bus [3..0]

Snapshot of RS (only 4 entries shown)

Adapted from Prof. G. Loh’s Slide
Selection Logic

Issue window

req

grant

anyreq

enable

Arbiter cell

enable
Simple (?) Select Logic Implementation

Reservation Station

Tree-like Arbitrated Selection Logic

[Palarchala ISCA’97]
Simple Select Logic Implementation

Reservation Station

Priority Decoder

AnyQueue
Enable

Req0
Grant0
Req1
Grant1
Req2
Grant2
Req3
Grant3

Enable
AnyQueue

AnyQueue
Enable

Req0
Grant0
Req1
Grant1
Req2
Grant2
Req3
Grant3
Simple Select Logic Implementation

Reservation Station

Multiple Ready Instruction Requests

[Palarchala ISCA'97]
Simple Select Logic Implementation

Reservation Station

Selective Issue for One FU
Issues to Distinctive Functional Units

Distributed Instruction Windows (e.g., MIPS R1000 or Alpha 21264)

Reservation Station

Faster to have separate instruction schedulers for different instruction
Dual Issues to Multiple Units
Bypass Delay

- The number of bypass paths equals $2 \times IW^2 \times S$ (S is the number of pipeline stages)

- Wire length $\sim IW$, hence, delay $\sim IW^2$

- The layout and pipeline depth (capacitive load) also matter
Other structures

- Registers
- Caches
Limitations of ILP

- Assume an ideal processor model
  - Register renaming
    - Infinite number of virtual registers
    - All WAW and WAR hazards can be removed
  - Branch prediction
    - Perfect branch prediction
  - Jump prediction
    - All jumps are perfectly predicted
  - Memory address alias analysis
    - All memory addresses are known exactly
  - Perfect caches
    - All memory accesses take 1 clock cycle
Impact of Window Size

Benchmark | Instruction issues per cycle
---|---
gcc | 10
     | 36
     | 55
espresso | 8
         | 15
         | 41
li | 9
fppp | 14
     | 49
doduc | 16
     | 59
      | 119
tomcatv | 14
        | 45
        | 60
        | 150

Window size
- Infinite
- 2K
- 512
- 128
- 32

© 2007 Elsevier, Inc. All rights reserved.
Impact of Branch Prediction

2K window
Max. 64 instruction issues per cycle

- gcc
  - Perfect: 6
  - Tournament predictor: 9
  - Standard 2-bit: 6
  - Profile-based: 2
  - None: 2

- espresso
  - Perfect: 7
  - Tournament predictor: 12
  - Standard 2-bit: 6
  - Profile-based: 2
  - None: 2

- li
  - Perfect: 6
  - Tournament predictor: 10
  - Standard 2-bit: 7
  - Profile-based: 2
  - None: 2

- fpppp
  - Perfect: 29
  - Tournament predictor: 48
  - Standard 2-bit: 46
  - Profile-based: 45
  - None: 45

- doduc
  - Perfect: 15
  - Tournament predictor: 48
  - Standard 2-bit: 46
  - Profile-based: 45
  - None: 45

- tomcatv
  - Perfect: 19
  - Tournament predictor: 48
  - Standard 2-bit: 46
  - Profile-based: 45
  - None: 45
Impact of Finite Registers

2K window
Max. 64 instruction issues per cycle
8K entry tournament predictors
2K jump and return predictors
Limits of Multiple-Issue Processors: Revisited

- Doubling the issue rate above the current 3-6 issue, i.e. 6-12 issue requires
  - Issue 3-4 data memory accesses per cycle
  - Resolve two or three branches per cycle
  - Rename and access more than 20 registers per cycle
  - Fetch 12-24 instructions per cycle
  - The complexity of implementing these capabilities would sacrifice the maximum clock rate

- Another issue is power!
  - Modern microprocessors are primarily power limited.
    - Static power grows proportionally to the transistor count
    - Dynamic power is proportional to the product of the number of switching transistors and the switching rate

- Microprocessors trying to achieve both a high IPC and a high CR must simultaneously increase clock speeds and transistor counts, leading to significant power consumption, which limits performance and energy efficiency.
Limits of Multiple-Issue Processors

- Energy inefficiency of multiple-issue processors
  - Multiple instruction issue incurs overhead in logic that grows faster than the issue rate increase
    - Dependence checking, register renaming, wakeup and select, etc.
    - Without voltage reduction, higher IPC will lead to lower performance/watt!
  - Growing gap between peak issue rate and sustained performance
    - The number of transistor switching is proportional to the peak issue rate
    - The performance is proportional to the sustained rate
    - Thus, growing gap translates to increasing energy per unit performance!
- For example, speculation is inherently inefficient
  - It can never be perfect, thus there is inherently waste in executing computations before we know that whether the path is taken or not
  - Increasing clock rate is also not energy efficient
Example: Pentium 4
A Superscalar CISC Machine
Pentium 4 alternate view

AGU = address generation unit
BTB = branch target buffer
D-TLB = data translation lookaside buffer
I-TLB = instruction translation lookaside buffer
Pentium 4 pipeline

20 stages!
HP PA-RISC 8000
Processor Performance with Time

Specint2000

Year of introduction
Price being paid

![Graph showing the relationship between Spec2000 and Watts/Spec for different processor types, including Intel, Alpha, Sparc, Mips, HP PA, Power PC, and AMD.](image)
4.58. **hsc - Halt and Spontaneously Combust**

<table>
<thead>
<tr>
<th>TY</th>
<th>OP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11111100000000000000000000000000000</td>
</tr>
</tbody>
</table>

**Assembler**

```
hsc
```

**Operation**

Reg(31) ← PC
The processor stops fetching instructions and self destructs.

Note that the contents of Reg(31) are actually lost.

**Description**

This is executed by the processor when a protection violation is detected. It is a privileged instruction available only on the -NSA versions of the processor.