Introduction to Code Optimization
Instruction Scheduling
Outline

• Modern architectures
• Introduction to instruction scheduling
• List scheduling
• Resource constraints
• Scheduling across basic blocks
• Trace scheduling
Simple Machine Model

- Instructions are executed in sequence
  - Fetch, decode, execute, store results
  - One instruction at a time

- For branch instructions, start fetching from a different location if needed
  - Check branch condition
  - Next instruction may come from a new location given by the branch instruction
Simple Execution Model

- 5 Stage pipe-line

<table>
<thead>
<tr>
<th>fetch</th>
<th>decode</th>
<th>execute</th>
<th>memory</th>
<th>writeback</th>
</tr>
</thead>
</table>

- Fetch: get the next instruction
- Decode: figure-out what that instruction is
- Execute: Perform ALU operation
  - address calculation in a memory op
- Write Back: write the results back
Simple Execution Model

Inst 1

IF | DE | EXE | MEM | WB

Inst 2

IF | DE | EXE | MEM | WB

Time arrow pointing from left to right.
Simple Execution Model

**Inst 1**

<table>
<thead>
<tr>
<th>IF</th>
<th>DE</th>
<th>EXE</th>
<th>MEM</th>
<th>WB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Inst 2**

<table>
<thead>
<tr>
<th>IF</th>
<th>DE</th>
<th>EXE</th>
<th>MEM</th>
<th>WB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Inst 3**

<table>
<thead>
<tr>
<th>IF</th>
<th>DE</th>
<th>EXE</th>
<th>MEM</th>
<th>WB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Inst 4**

<table>
<thead>
<tr>
<th>IF</th>
<th>DE</th>
<th>EXE</th>
<th>MEM</th>
<th>WB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Inst 5**

<table>
<thead>
<tr>
<th>IF</th>
<th>DE</th>
<th>EXE</th>
<th>MEM</th>
<th>WB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Outline

• Modern architectures
• Introduction to instruction scheduling
• List scheduling
• Resource constraints
• Scheduling across basic blocks
• Trace scheduling
From a Simple Machine Model to a Real Machine Model

• Many pipeline stages
  - Pentium 5
  - Pentium Pro 10
  - Pentium IV (130nm) 20
  - Pentium IV (90nm) 31
  - Core 2 Duo 14

• Different instructions taking different amount of time to execute

• Hardware to stall the pipeline if an instruction uses a result that is not ready
Real Machine Model cont.

- Most modern processors have multiple cores

- Each core has multiple execution units (superscalar)
  - If the instruction sequence is efficient, multiple operations will happen in the same cycles
  - Even more important to have the right instruction sequence
Constraints On Scheduling

- Data dependencies
- Control dependencies
- Resource Constraints

- **Goal:** Reorder instructions so that pipeline stalls are minimized
Data Dependency between Instructions

• If two instructions access the same variable, they can be dependent

• Kind of dependencies
  – True: write → read
  – Anti: read → write
  – Output: write → write

• What to do if two instructions are dependent.
  – The order of execution cannot be reversed
  – Reduce the possibilities for scheduling
Computing Dependencies

• For basic blocks, compute dependencies by walking through the instructions

• Identifying register dependencies is simple
  – is it the same register?

• For memory accesses
  – simple: base + offset1 \(\neq\) base + offset2
  – data dependence analysis: \(a[2i] \neq a[2i+1]\)
  – interprocedural analysis: \(\text{global} \neq \text{parameter}\)
  – pointer alias analysis: \(p1 \rightarrow \text{foo} \neq p2 \rightarrow \text{foo}\)
Representing Dependencies

• Using a dependence DAG, one per basic block
• Nodes are instructions, edges represent dependencies
Representing Dependencies

- Using a dependence DAG, one per basic block
- Nodes are instructions, edges represent dependencies

1: \( r_2 = *(r_1 + 4) \)
2: \( r_3 = *(r_1 + 8) \)
3: \( r_4 = r_2 + r_3 \)
4: \( r_5 = r_2 - 1 \)
Representing Dependencies

- Using a dependence DAG, one per basic block
- Nodes are instructions, edges represent dependencies

1: \( r_2 = *(r_1 + 4) \)
2: \( r_3 = *(r_1 + 8) \)
3: \( r_4 = r_2 + r_3 \)
4: \( r_5 = r_2 - 1 \)
Representing Dependencies

- Using a dependence DAG, one per basic block
- Nodes are instructions, edges represent dependencies

1: \( r_2 = *(r_1 + 4) \)
2: \( r_3 = *(r_1 + 8) \)
3: \( r_4 = r_2 + r_3 \)
4: \( r_5 = r_2 - 1 \)

- Edge is labeled with Latency:
  \[ v(i \rightarrow j) = \text{delay required between initiation times of } i \text{ and } j \text{ minus the execution time required by } i \]
Control Dependencies and Resource Constraints

- For now, let's only worry about basic blocks
- For now, let's look at simple pipelines
Example

1: lea var_a, %rax
2: add $4, %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: add %r10, 8(%rsp)
6: and 16(%rsp), %rbx
7: imul %rax, %rbx
### Example

<table>
<thead>
<tr>
<th>Line</th>
<th>Instruction</th>
<th>Results In</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>lea  var_a, %rax</td>
<td>1 cycle</td>
</tr>
<tr>
<td>2</td>
<td>add $4, %rax</td>
<td>1 cycle</td>
</tr>
<tr>
<td>3</td>
<td>inc %r11</td>
<td>1 cycle</td>
</tr>
<tr>
<td>4</td>
<td>mov  4(%rsp), %r10</td>
<td>3 cycles</td>
</tr>
<tr>
<td>5</td>
<td>add  %r10, 8(%rsp)</td>
<td>3 cycles</td>
</tr>
<tr>
<td>6</td>
<td>and  16(%rsp), %rbx</td>
<td>4 cycles</td>
</tr>
<tr>
<td>7</td>
<td>imul %rax, %rbx</td>
<td>3 cycles</td>
</tr>
</tbody>
</table>
## Example

<table>
<thead>
<tr>
<th>Line</th>
<th>Instruction</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>lea var_a, %rax</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>add $4, %rax</td>
<td>1 cycle</td>
</tr>
<tr>
<td>3</td>
<td>inc %r11</td>
<td>1 cycle</td>
</tr>
<tr>
<td>4</td>
<td>mov 4(%rsp), %r10</td>
<td>3 cycles</td>
</tr>
<tr>
<td>5</td>
<td>add %r10, 8(%rsp)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>and 16(%rsp), %rbx</td>
<td>4 cycles</td>
</tr>
<tr>
<td>7</td>
<td>imul %rax, %rbx</td>
<td>3 cycles</td>
</tr>
</tbody>
</table>
## Example

<table>
<thead>
<tr>
<th></th>
<th>Instruction</th>
<th>Results In</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>lea var_a, %rax</td>
<td>1 cycle</td>
</tr>
<tr>
<td>2</td>
<td>add $4, %rax</td>
<td>1 cycle</td>
</tr>
<tr>
<td>3</td>
<td>inc %r11</td>
<td>1 cycle</td>
</tr>
<tr>
<td>4</td>
<td>mov 4(%rsp), %r10</td>
<td>3 cycles</td>
</tr>
<tr>
<td>5</td>
<td>add %r10, 8(%rsp)</td>
<td>3 cycles</td>
</tr>
<tr>
<td>6</td>
<td>and 16(%rsp), %rbx</td>
<td>4 cycles</td>
</tr>
<tr>
<td>7</td>
<td>imul %rax, %rbx</td>
<td>3 cycles</td>
</tr>
</tbody>
</table>
## Example

<table>
<thead>
<tr>
<th></th>
<th>Instruction</th>
<th>Results In</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>lea var_a, %rax</td>
<td>1 cycle</td>
</tr>
<tr>
<td>2</td>
<td>add $4, %rax</td>
<td>1 cycle</td>
</tr>
<tr>
<td>3</td>
<td>inc %r11</td>
<td>1 cycle</td>
</tr>
<tr>
<td>4</td>
<td>mov 4(%rsp), %r10</td>
<td>3 cycles</td>
</tr>
<tr>
<td>5</td>
<td>add %r10, 8(%rsp)</td>
<td>3 cycles</td>
</tr>
<tr>
<td>6</td>
<td>and 16(%rsp), %rbx</td>
<td>4 cycles</td>
</tr>
<tr>
<td>7</td>
<td>imul %rax, %rbx</td>
<td>3 cycles</td>
</tr>
</tbody>
</table>
Example

1: lea var_a, %rax
2: add $4, %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: add %r10, 8(%rsp)
6: and 16(%rsp), %rbx
7: imul %rax, %rbx

Results In
1 cycle
1 cycle
1 cycle
3 cycles
3 cycles
4 cycles
3 cycles
Example

1: lea  var_a, %rax
2: add  $4, %rax
3: inc  %r11
4: mov  4(%rsp), %r10
5: add  %r10, 8(%rsp)
6: and  16(%rsp), %rbx
7: imul %rax, %rbx

Results In
1 cycle
1 cycle
1 cycle
3 cycles
3 cycles
4 cycles
3 cycles
Example

<table>
<thead>
<tr>
<th></th>
<th>Instruction</th>
<th>Results In</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>lea var_a, %rax</td>
<td>1 cycle</td>
</tr>
<tr>
<td>2</td>
<td>add $4, %rax</td>
<td>1 cycle</td>
</tr>
<tr>
<td>3</td>
<td>inc %r11</td>
<td>1 cycle</td>
</tr>
<tr>
<td>4</td>
<td>mov 4(%rsp), %r10</td>
<td>3 cycles</td>
</tr>
<tr>
<td>5</td>
<td>add %r10, 8(%rsp)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>and 16(%rsp), %rbx</td>
<td>4 cycles</td>
</tr>
<tr>
<td>7</td>
<td>imul %rax, %rbx</td>
<td>3 cycles</td>
</tr>
</tbody>
</table>
### Example

<table>
<thead>
<tr>
<th></th>
<th>Instruction</th>
<th>Results In</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>lea var_a, %rax</td>
<td>1 cycle</td>
</tr>
<tr>
<td>2</td>
<td>add $4, %rax</td>
<td>1 cycle</td>
</tr>
<tr>
<td>3</td>
<td>inc %r11</td>
<td>1 cycle</td>
</tr>
<tr>
<td>4</td>
<td>mov 4(%rsp), %r10</td>
<td>3 cycles</td>
</tr>
<tr>
<td>5</td>
<td>add %r10, 8(%rsp)</td>
<td>3 cycles</td>
</tr>
<tr>
<td>6</td>
<td>and 16(%rsp), %rbx</td>
<td>4 cycles</td>
</tr>
<tr>
<td>7</td>
<td>imul %rax, %rbx</td>
<td>3 cycles</td>
</tr>
</tbody>
</table>
Example

1: lea var_a, %rax
2: add $4, %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: add %r10, 8(%rsp)
6: and 16(%rsp), %rbx
7: imul %rax, %rbx

Results In
1 cycle
1 cycle
1 cycle
3 cycles
3 cycles
4 cycles
3 cycles
### Example

<table>
<thead>
<tr>
<th></th>
<th>Instruction</th>
<th>Results In</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>lea var_a, %rax</td>
<td>1 cycle</td>
</tr>
<tr>
<td>2</td>
<td>add $4, %rax</td>
<td>1 cycle</td>
</tr>
<tr>
<td>3</td>
<td>inc %r11</td>
<td>1 cycle</td>
</tr>
<tr>
<td>4</td>
<td>mov 4(%rsp), %r10</td>
<td>3 cycles</td>
</tr>
<tr>
<td>5</td>
<td>add %r10, 8(%rsp)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>and 16(%rsp), %rbx</td>
<td>4 cycles</td>
</tr>
<tr>
<td>7</td>
<td>imul %rax, %rbx</td>
<td>3 cycles</td>
</tr>
</tbody>
</table>
### Example

<table>
<thead>
<tr>
<th></th>
<th>Command</th>
<th>Results In</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>lea     var_a, %rax</td>
<td>1 cycle</td>
</tr>
<tr>
<td>2</td>
<td>add     $4, %rax</td>
<td>1 cycle</td>
</tr>
<tr>
<td>3</td>
<td>inc      %r11</td>
<td>1 cycle</td>
</tr>
<tr>
<td>4</td>
<td>mov      4(%rsp), %r10</td>
<td>3 cycles</td>
</tr>
<tr>
<td>5</td>
<td>add      %r10, 8(%rsp)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>and      16(%rsp), %rbx</td>
<td>4 cycles</td>
</tr>
<tr>
<td>7</td>
<td>imul     %rax, %rbx</td>
<td>3 cycles</td>
</tr>
</tbody>
</table>
### Example

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Results In</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: lea var_a, %rax</td>
<td>1 cycle</td>
</tr>
<tr>
<td>2: add $4, %rax</td>
<td>1 cycle</td>
</tr>
<tr>
<td>3: inc %r11</td>
<td>1 cycle</td>
</tr>
<tr>
<td>4: mov 4(%rsp), %r10</td>
<td>3 cycles</td>
</tr>
<tr>
<td>5: add %r10, 8(%rsp)</td>
<td>4 cycles</td>
</tr>
<tr>
<td>6: and 16(%rsp), %rbx</td>
<td>3 cycles</td>
</tr>
<tr>
<td>7: imul %rax, %rbx</td>
<td>3 cycles</td>
</tr>
</tbody>
</table>
### Example

<table>
<thead>
<tr>
<th></th>
<th>Instruction</th>
<th></th>
<th>Results In</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>lea var_a, %rax</td>
<td>1</td>
<td>1 cycle</td>
</tr>
<tr>
<td>2</td>
<td>add $4, %rax</td>
<td>1</td>
<td>1 cycle</td>
</tr>
<tr>
<td>3</td>
<td>inc %r11</td>
<td>1</td>
<td>1 cycle</td>
</tr>
<tr>
<td>4</td>
<td>mov 4(%rsp), %r10</td>
<td>3</td>
<td>3 cycles</td>
</tr>
<tr>
<td>5</td>
<td>add %r10, 8(%rsp)</td>
<td>4</td>
<td>4 cycles</td>
</tr>
<tr>
<td>6</td>
<td>and 16(%rsp), %rbx</td>
<td></td>
<td>3 cycles</td>
</tr>
<tr>
<td>7</td>
<td>imul %rax, %rbx</td>
<td></td>
<td>3 cycles</td>
</tr>
</tbody>
</table>
## Example

<table>
<thead>
<tr>
<th></th>
<th>Instruction</th>
<th></th>
<th>Results In</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>lea var_a, %rax</code></td>
<td></td>
<td>1 cycle</td>
</tr>
<tr>
<td>2</td>
<td><code>add $4, %rax</code></td>
<td></td>
<td>1 cycle</td>
</tr>
<tr>
<td>3</td>
<td><code>inc %r11</code></td>
<td></td>
<td>1 cycle</td>
</tr>
<tr>
<td>4</td>
<td><code>mov 4(%rsp), %r10</code></td>
<td></td>
<td>3 cycles</td>
</tr>
<tr>
<td>5</td>
<td><code>add %r10, 8(%rsp)</code></td>
<td></td>
<td>3 cycles</td>
</tr>
<tr>
<td>6</td>
<td><code>and 16(%rsp), %rbx</code></td>
<td></td>
<td>4 cycles</td>
</tr>
<tr>
<td>7</td>
<td><code>imul %rax, %rbx</code></td>
<td></td>
<td>3 cycles</td>
</tr>
</tbody>
</table>
### Example

<table>
<thead>
<tr>
<th>Step</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>lea var_a, %rax</td>
</tr>
<tr>
<td>2</td>
<td>add $4, %rax</td>
</tr>
<tr>
<td>3</td>
<td>inc %r11</td>
</tr>
<tr>
<td>4</td>
<td>mov 4(%rsp), %r10</td>
</tr>
<tr>
<td>5</td>
<td>add %r10, 8(%rsp)</td>
</tr>
<tr>
<td>6</td>
<td>and 16(%rsp), %rbx</td>
</tr>
<tr>
<td>7</td>
<td>imul %rax, %rbx</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Results In</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 cycle</td>
</tr>
<tr>
<td>1 cycle</td>
</tr>
<tr>
<td>1 cycle</td>
</tr>
<tr>
<td>3 cycles</td>
</tr>
<tr>
<td>3 cycles</td>
</tr>
<tr>
<td>4 cycles</td>
</tr>
<tr>
<td>3 cycles</td>
</tr>
</tbody>
</table>
Example

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>lea</td>
<td>var_a, %rax</td>
<td></td>
<td></td>
<td></td>
<td>Results In 1 cycle</td>
</tr>
<tr>
<td>2</td>
<td>add</td>
<td>$4, %rax</td>
<td></td>
<td></td>
<td></td>
<td>1 cycle</td>
</tr>
<tr>
<td>3</td>
<td>inc</td>
<td>%r11</td>
<td></td>
<td></td>
<td></td>
<td>1 cycle</td>
</tr>
<tr>
<td>4</td>
<td>mov</td>
<td>4(%rsp), %r10</td>
<td></td>
<td></td>
<td></td>
<td>3 cycles</td>
</tr>
<tr>
<td>5</td>
<td>add</td>
<td>%r10, 8(%rsp)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>and</td>
<td>16(%rsp), %rbx</td>
<td></td>
<td></td>
<td></td>
<td>4 cycles</td>
</tr>
<tr>
<td>7</td>
<td>imul</td>
<td>%rax, %rbx</td>
<td></td>
<td></td>
<td></td>
<td>3 cycles</td>
</tr>
</tbody>
</table>
Example

<table>
<thead>
<tr>
<th></th>
<th>Assembly Instruction</th>
<th>Results In</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>lea var_a, %rax</code></td>
<td>1 cycle</td>
</tr>
<tr>
<td>2</td>
<td><code>add $4, %rax</code></td>
<td>1 cycle</td>
</tr>
<tr>
<td>3</td>
<td><code>inc %r11</code></td>
<td>1 cycle</td>
</tr>
<tr>
<td>4</td>
<td><code>mov 4(%rsp), %r10</code></td>
<td>3 cycles</td>
</tr>
<tr>
<td>5</td>
<td><code>add %r10, 8(%rsp)</code></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td><code>and 16(%rsp), %rbx</code></td>
<td>4 cycles</td>
</tr>
<tr>
<td>7</td>
<td><code>imul %rax, %rbx</code></td>
<td>3 cycles</td>
</tr>
</tbody>
</table>
### Example

<table>
<thead>
<tr>
<th></th>
<th>Instruction</th>
<th>Results In</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>lea var_a, %rax</code></td>
<td>1 cycle</td>
</tr>
<tr>
<td>2</td>
<td><code>add $4, %rax</code></td>
<td>1 cycle</td>
</tr>
<tr>
<td>3</td>
<td><code>inc %r11</code></td>
<td>1 cycle</td>
</tr>
<tr>
<td>4</td>
<td><code>mov 4(%rsp), %r10</code></td>
<td>3 cycles</td>
</tr>
<tr>
<td>5</td>
<td><code>add %r10, 8(%rsp)</code></td>
<td>3 cycles</td>
</tr>
<tr>
<td>6</td>
<td><code>and 16(%rsp), %rbx</code></td>
<td>4 cycles</td>
</tr>
<tr>
<td>7</td>
<td><code>imul %rax, %rbx</code></td>
<td>3 cycles</td>
</tr>
</tbody>
</table>
# Example

1: lea var_a, %rax  
2: add $4, %rax  
3: inc %r11  
4: mov 4(%rsp), %r10  
5: add %r10, 8(%rsp)  
6: and 16(%rsp), %rbx  
7: imul %rax, %rbx

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Results In</th>
</tr>
</thead>
<tbody>
<tr>
<td>lea var_a, %rax</td>
<td>1 cycle</td>
</tr>
<tr>
<td>add $4, %rax</td>
<td>1 cycle</td>
</tr>
<tr>
<td>inc %r11</td>
<td>1 cycle</td>
</tr>
<tr>
<td>mov 4(%rsp), %r10</td>
<td>3 cycles</td>
</tr>
<tr>
<td>add %r10, 8(%rsp)</td>
<td>3 cycles</td>
</tr>
<tr>
<td>and 16(%rsp), %rbx</td>
<td>4 cycles</td>
</tr>
<tr>
<td>imul %rax, %rbx</td>
<td>3 cycles</td>
</tr>
</tbody>
</table>
### Example

<table>
<thead>
<tr>
<th></th>
<th>Instruction</th>
<th>Results In</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>lea var_a, %rax</td>
<td>1 cycle</td>
</tr>
<tr>
<td>2</td>
<td>add $4, %rax</td>
<td>1 cycle</td>
</tr>
<tr>
<td>3</td>
<td>inc %r11</td>
<td>1 cycle</td>
</tr>
<tr>
<td>4</td>
<td>mov 4(%rsp), %r10</td>
<td>3 cycles</td>
</tr>
<tr>
<td>5</td>
<td>add %r10, 8(%rsp)</td>
<td>3 cycles</td>
</tr>
<tr>
<td>6</td>
<td>and 16(%rsp), %rbx</td>
<td>4 cycles</td>
</tr>
<tr>
<td>7</td>
<td>imul %rax, %rbx</td>
<td>3 cycles</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>1</td>
<td>lea</td>
<td>var_a, %rax</td>
</tr>
<tr>
<td>2</td>
<td>add</td>
<td>$4, %rax</td>
</tr>
<tr>
<td>3</td>
<td>inc</td>
<td>%r11</td>
</tr>
<tr>
<td>4</td>
<td>mov</td>
<td>4(%rsp), %r10</td>
</tr>
<tr>
<td>5</td>
<td>add</td>
<td>%r10, 8(%rsp)</td>
</tr>
<tr>
<td>6</td>
<td>and</td>
<td>16(%rsp), %rbx</td>
</tr>
<tr>
<td>7</td>
<td>imul</td>
<td>%rax, %rbx</td>
</tr>
</tbody>
</table>

Example
Outline

• Modern architectures
• Introduction to instruction scheduling
• List scheduling
• Resource constraints
• Scheduling across basic blocks
• Trace scheduling
List Scheduling Algorithm

• Idea
  – Do a topological sort of the dependence DAG
  – Consider when an instruction can be scheduled without causing a stall
  – Schedule the instruction if it causes no stall and all its predecessors are already scheduled

• Optimal list scheduling is NP-complete
  – Use heuristics when necessary
List Scheduling Algorithm

• Create a dependence DAG of a basic block
• Topological Sort
  READY = nodes with no predecessors
  Loop until READY is empty
    Schedule each node in READY when no stalling
  Update READY
Heuristics for selection

- Heuristics for selecting from the READY list
  - pick the node with the longest path to a leaf in the dependence graph
  - pick a node with most immediate successors
  - pick a node that can go to a less busy pipeline (in a superscalar)
Heuristics for selection

• Pick the node with the longest path to a leaf in the dependence graph

• Algorithm (for node x)
  – If no successors \( d_x = 0 \)
  – \( d_x = \text{MAX}(d_y + c_{xy}) \) for all successors y of x
  – reverse breadth-first visitation order
Heuristics for selection

• Pick a node with most immediate successors
• Algorithm (for node x):
  \[ f_x = \text{number of successors of } x \]
Example

1: lea var_a, %rax
2: add $4, %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: add %r10, 8(%rsp)
6: and 16(%rsp), %rbx
7: imul %rax, %rbx
8: mov %rbx, 16(%rsp)
9: lea var_b, %rax
Example

1: lea var_a, %rax
2: add $4, %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: add %r10, 8(%rsp)
6: and 16(%rsp), %rbx
7: imul %rax, %rbx
8: mov %rbx, 16(%rsp)
9: lea var_b, %rax

Results In
1 cycle
1 cycle
1 cycle
3 cycles
3 cycles
4 cycles
3 cycles
Example

1: lea var_a, %rax
2: add $4, %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: add %r10, 8(%rsp)
6: and 16(%rsp), %rbx
7: imul %rax, %rbx
8: mov %rbx, 16(%rsp)
9: lea var_b, %rax
1: lea var_a, %rax
2: add $4, %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: add %r10, 8(%rsp)
6: and 16(%rsp), %rbx
7: imul %rax, %rbx
8: mov %rbx, 16(%rsp)
9: lea var_b, %rax
1: lea var_a, %rax
2: add $4, %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: add %r10, 8(%rsp)
6: and 16(%rsp), %rbx
7: imul %rax, %rbx
8: mov %rbx, 16(%rsp)
9: lea var_b, %rax
Example

1: lea var_a, %rax
2: add $4, %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: add %r10, 8(%rsp)
6: and 16(%rsp), %rbx
7: imul %rax, %rbx
8: mov %rbx, 16(%rsp)
9: lea var_b, %rax
1: lea var_a, %rax
2: add $4, %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: add %r10, 8(%rsp)
6: and 16(%rsp), %rbx
7: imul %rax, %rbx
8: mov %rbx, 16(%rsp)
9: lea var_b, %rax
Example

1: lea var_a, %rax
2: add $4, %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: add %r10, 8(%rsp)
6: and 16(%rsp), %rbx
7: imul %rax, %rbx
8: mov %rbx, 16(%rsp)
9: lea var_b, %rax
1: lea var_a, %rax
2: add $4, %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: add %r10, 8(%rsp)
6: and 16(%rsp), %rbx
7: imul %rax, %rbx
8: mov %rbx, 16(%rsp)
9: lea var_b, %rax
Example

1: lea var_a, %rax
2: add $4, %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: add %r10, 8(%rsp)
6: and 16(%rsp), %rbx
7: imul %rax, %rbx
8: mov %rbx, 16(%rsp)
9: lea var_b, %rax
1: lea var_a, %rax
2: add $4, %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: add %r10, 8(%rsp)
6: and 16(%rsp), %rbx
7: imul %rax, %rbx
8: mov %rbx, 16(%rsp)
9: lea var_b, %rax
Example

\[ \text{READY} = \{ \} \]
Example

1, 3, 4, 6

READY = {}
Example

1, 3, 4, 6

READY = \{ 6, 1, 4, 3 \}
READY = \{ 6, 1, 4, 3 \}
READY = \{ 6, 1, 4, 3 \}
READY = \{ 1, 4, 3 \}
Example

\[
\text{READY} = \{ 1, 4, 3 \}
\]
Example

$\text{READY} = \{1, 4, 3\}$
Example

READY = \{ 4, 3 \}
Example

READY = \{ 2, 4, 3 \}
Example

READY = \{ 2, 4, 3 \}
Example

READY = \{ 2, 4, 3 \}
Example

READY = { 2, 4, 3 }
Example

\[ \text{READY} = \{ 4, 3 \} \]
READY = \{ 7, 4, 3 \}
Example

READY = \{ 7, 4, 3 \}
Example

READY = \{ 7, 4, 3 \}
Example

\[ \text{READY} = \{ 7, 4, 3 \} \]
Example

READY = \{ 7, 4, 3 \}
Example

$\text{READY} = \{7, 3\}$
READY = \{ 7, 3, 5 \}
Example

\[
\text{READY} = \{7, 3, 5\}
\]
Example

\[
\text{READY} = \{ 7, 3, 5 \}
\]
Example

READY = \{ 7, 3, 5 \}
Example

READY = \{ 3, 5 \}

8, 9
Example

$\text{READY} = \{3, 5, 8, 9\}$
Example

READY = \{ 3, 5, 8, 9 \}
Example

READY = { 3, 5, 8, 9 }
READY = \{ 5, 8, 9 \}
Example

\[ \text{READY} = \{ 5, 8, 9 \} \]
Example

READY = { 5, 8, 9 }
Example

READY = \{ 5, 8, 9 \}
READY = \{ 8, 9 \}
Example

READY = { 8, 9 }
Example

READY = \{ 8, 9 \}
Example

$\text{READY} = \{ 8, 9 \}$
READY = \{ 9 \}
Example

READY = { 9 }
Example

\[
\text{READY} = \{9\}
\]
Example

READY = { 9 }

6 1 2 4 7 3 5 8 9
Example

READY = \{ \}
**Example**

<table>
<thead>
<tr>
<th>Line</th>
<th>Assembly Code</th>
<th>Results In</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>lea var_a, %rax</td>
<td>1 cycle</td>
</tr>
<tr>
<td>2</td>
<td>add $4, %rax</td>
<td>1 cycle</td>
</tr>
<tr>
<td>3</td>
<td>inc %r11</td>
<td>1 cycle</td>
</tr>
<tr>
<td>4</td>
<td>mov 4(%rsp), %r10</td>
<td>3 cycles</td>
</tr>
<tr>
<td>5</td>
<td>add %r10, 8(%rsp)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>and 16(%rsp), %rbx</td>
<td>4 cycles</td>
</tr>
<tr>
<td>7</td>
<td>imul %rax, %rbx</td>
<td>3 cycles</td>
</tr>
<tr>
<td>8</td>
<td>mov %rbx, 16(%rsp)</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>lea var_b, %rax</td>
<td></td>
</tr>
</tbody>
</table>

*9 cycles*
### Example

<table>
<thead>
<tr>
<th>Line</th>
<th>Instruction</th>
<th>Results In</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>lea var_a, %rax</td>
<td>1 cycle</td>
</tr>
<tr>
<td>2</td>
<td>add $4, %rax</td>
<td>1 cycle</td>
</tr>
<tr>
<td>3</td>
<td>inc %r11</td>
<td>1 cycle</td>
</tr>
<tr>
<td>4</td>
<td>mov 4(%rsp), %r10</td>
<td>3 cycles</td>
</tr>
<tr>
<td>5</td>
<td>add %r10, 8(%rsp)</td>
<td>3 cycles</td>
</tr>
<tr>
<td>6</td>
<td>and 16(%rsp), %rbx</td>
<td>4 cycles</td>
</tr>
<tr>
<td>7</td>
<td>imul %rax, %rbx</td>
<td>3 cycles</td>
</tr>
<tr>
<td>8</td>
<td>mov %rbx, 16(%rsp)</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>lea var_b, %rax</td>
<td></td>
</tr>
</tbody>
</table>

**14 cycles vs 9 cycles**
Outline

• Modern architectures
• Introduction to instruction scheduling
• List scheduling
• Resource constraints
• Scheduling across basic blocks
• Trace scheduling
Resource Constraints

• Modern machines have many resource constraints

• Superscalar architectures:
  – can run a few parallel operations
  – But have constraints
Resource Constraints of a Superscalar Processor

• Example:
  – One fully pipelined reg-to-reg unit
    • All integer operations taking one cycle
  In parallel with
  – One fully pipelined memory-to/from-reg unit
    • Data loads take two cycles
    • Data stores take one cycle
List Scheduling Algorithm with resource constraints

- Represent the superscalar architecture as multiple pipelines
  - Each pipeline represents some resource
List Scheduling Algorithm with resource constraints

• Represent the superscalar architecture as multiple pipelines
  – Each pipeline represent some resource

• Example
  – One single cycle reg-to-reg ALU unit
  – One two-cycle pipelined reg-to/from-memory unit
List Scheduling Algorithm with resource constraints

- Create a dependence DAG of a basic block
- Topological Sort
  
  READY = nodes with no predecessors
  
  Loop until READY is empty
    
    Let \( n \in \text{READY} \) be the node with the highest priority
    
    Schedule \( n \) in the earliest slot that satisfies precedence + resource constraints
    
    Update READY
Example

1: lea var_a, %rax
2: add 4(%rsp), %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: mov %r10, 8(%rsp)
6: and $0x00ff, %rbx
7: imul %rax, %rbx
8: lea var_b, %rax
9: mov %rbx, 16(%rsp)
Example

1: lea var_a, %rax
2: add 4(%rsp), %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: mov %r10, 8(%rsp)
6: and $0x00ff, %rbx
7: imul %rax, %rbx
8: lea var_b, %rax
9: mov %rbx, 16(%rsp)
Example

1: lea var_a, %rax
2: add 4(%rsp), %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: mov %r10, 8(%rsp)
6: and $0x00ff, %rbx
7: imul %rax, %rbx
8: lea var_b, %rax
9: mov %rbx, 16(%rsp)
Example

1: lea var_a, %rax
2: add 4(%rsp), %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: mov %r10, 8(%rsp)
6: and $0x00ff, %rbx
7: imul %rax, %rbx
8: lea var_b, %rax
9: mov %rbx, 16(%rsp)

READY = { 1, 6, 4, 3 }
Example

1: lea var_a, %rax
2: add 4(%rsp), %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: mov %r10, 8(%rsp)
6: and $0x00ff, %rbx
7: imul %rax, %rbx
8: lea var_b, %rax
9: mov %rbx, 16(%rsp)

READY = { 1, 6, 4, 3 }

ALUop
MEM 1
MEM 2
Example

1: lea var_a, %rax
2: add 4(%rsp), %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: mov %r10, 8(%rsp)
6: and $0x00ff, %rbx
7: imul %rax, %rbx
8: lea var_b, %rax
9: mov %rbx, 16(%rsp)

READY = { 1, 6, 4, 3 }
Example

1: lea var_a, %rax
2: add 4(%rsp), %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: mov %r10, 8(%rsp)
6: and $0x00ff, %rbx
7: imul %rax, %rbx
8: lea var_b, %rax
9: mov %rbx, 16(%rsp)

READY = { 1, 6, 4, 3 }
Example

1: lea var_a, %rax
2: add 4(%rsp), %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: mov %r10, 8(%rsp)
6: and $0x00ff, %rbx
7: imul %rax, %rbx
8: lea var_b, %rax
9: mov %rbx, 16(%rsp)

READY = { 6, 4, 3 } ← 2
Example

1: lea var_a, %rax
2: add 4(%rsp), %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: mov %r10, 8(%rsp)
6: and $0x00ff, %rbx
7: imul %rax, %rbx
8: lea var_b, %rax
9: mov %rbx, 16(%rsp)

READY = \{ 2, 6, 4, 3 \}

<table>
<thead>
<tr>
<th>ALUop</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEM 1</td>
<td></td>
</tr>
<tr>
<td>MEM 2</td>
<td></td>
</tr>
</tbody>
</table>
Example

1: lea var_a, %rax
2: add 4(%rsp), %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: mov %r10, 8(%rsp)
6: and $0x00ff, %rbx
7: imul %rax, %rbx
8: lea var_b, %rax
9: mov %rbx, 16(%rsp)

READY = { 2, 6, 4, 3 }
Example

1: lea var_a, %rax
2: add 4(%rsp), %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: mov %r10, 8(%rsp)
6: and $0x00ff, %rbx
7: imul %rax, %rbx
8: lea var_b, %rax
9: mov %rbx, 16(%rsp)

READY = { 2, 6, 4, 3 }

<table>
<thead>
<tr>
<th>ALUop</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEM 1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>MEM 2</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
Example

1: lea var_a, %rax
2: add 4(%rsp), %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: mov %r10, 8(%rsp)
6: and $0x00ff, %rbx
7: imul %rax, %rbx
8: lea var_b, %rax
9: mov %rbx, 16(%rsp)

READY = { 6, 4, 3 }

<table>
<thead>
<tr>
<th>ALUop</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEM 1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>MEM 2</td>
<td></td>
<td>2</td>
</tr>
</tbody>
</table>
Example

1: lea var_a, %rax
2: add 4(%.rsp), %rax
3: inc %r11
4: mov 4(%.rsp), %r10
5: mov %r10, 8(%.rsp)
6: and $0x00ff, %rbx
7: imul %rax, %rbx
8: lea var_b, %rax
9: mov %rbx, 16(%.rsp)

READY = { 6, 4, 3 }

<table>
<thead>
<tr>
<th>ALUop</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEM 1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>MEM 2</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
Example

1: lea var_a, %rax
2: add 4(%rsp), %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: mov %r10, 8(%rsp)
6: and $0x00ff, %rbx
7: imul %rax, %rbx
8: lea var_b, %rax
9: mov %rbx, 16(%rsp)

READY = { 6, 4, 3 }

<table>
<thead>
<tr>
<th>ALUop</th>
<th>1</th>
<th>6</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEM 1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MEM 2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example

1: lea var_a, %rax
2: add 4(%rsp), %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: mov %r10, 8(%rsp)
6: and $0x00ff, %rbx
7: imul %rax, %rbx
8: lea var_b, %rax
9: mov %rbx, 16(%rsp)

READY = { 4, 3 } ← 7

<table>
<thead>
<tr>
<th>ALUop</th>
<th>1</th>
<th>6</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEM 1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MEM 2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example

1: lea var_a, %rax  
2: add 4(%rsp), %rax  
3: inc %r11  
4: mov 4(%rsp), %r10  
5: mov %r10, 8(%rsp)  
6: and $0x00ff, %rbx  
7: imul %rax, %rbx  
8: lea var_b, %rax  
9: mov %rbx, 16(%rsp)

READY = { 4, 7, 3 }  

<table>
<thead>
<tr>
<th>ALUop</th>
<th>1</th>
<th>6</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEM 1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MEM 2</td>
<td></td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
Example

1: lea var_a, %rax
2: add 4(%rsp), %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: mov %r10, 8(%rsp)
6: and $0x00ff, %rbx
7: imul %rax, %rbx
8: lea var_b, %rax
9: mov %rbx, 16(%rsp)

READY = { 4, 7, 3 }

<table>
<thead>
<tr>
<th>ALUop</th>
<th>1</th>
<th>6</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEM 1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MEM 2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example

1: lea var_a, %rax
2: add 4(%rsp), %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: mov %r10, 8(%rsp)
6: and $0x00ff, %rbx
7: imul %rax, %rbx
8: lea var_b, %rax
9: mov %rbx, 16(%rsp)

READY = { 4, 7, 3 }

<table>
<thead>
<tr>
<th>ALUop</th>
<th>1</th>
<th>6</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEM 1</td>
<td>4</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>MEM 2</td>
<td>4</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
Example

1: lea var_a, %rax
2: add 4(%rsp), %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: mov %r10, 8(%rsp)
6: and $0x00ff, %rbx
7: imul %rax, %rbx
8: lea var_b, %rax
9: mov %rbx, 16(%rsp)

READY = { 7, 3 } ← 5

<table>
<thead>
<tr>
<th>ALUop</th>
<th>1</th>
<th>6</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEM 1</td>
<td>4</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>MEM 2</td>
<td>4</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
Example

1: \texttt{lea} \ var\_a, \ %rax
2: \texttt{add} \ 4(\%rsp), \ %rax
3: \texttt{inc} \ %r11
4: \texttt{mov} \ 4(\%rsp), \ %r10
5: \texttt{mov} \ %r10, \ 8(\%rsp)
6: \texttt{and} \ $0\times00ff, \ %rbx
7: \texttt{imul} \ %rax, \ %rbx
8: \texttt{lea} \ \texttt{var}_b, \ %rax
9: \texttt{mov} \ %rbx, \ 16(\%rsp)

\textbf{READY} = \{7, 3, 5\}

<table>
<thead>
<tr>
<th>ALUop</th>
<th>1</th>
<th>6</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEM 1</td>
<td>4</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>MEM 2</td>
<td>4</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
Example

1: lea var_a, %rax
2: add 4(%rsp), %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: mov %r10, 8(%rsp)
6: and $0x00ff, %rbx
7: imul %rax, %rbx
8: lea var_b, %rax
9: mov %rbx, 16(%rsp)

READY = { 7, 3, 5 }

<table>
<thead>
<tr>
<th>ALUop</th>
<th>1</th>
<th>6</th>
<th>2</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEM 1</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MEM 2</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example

1: lea var_a, %rax
2: add 4(%rsp), %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: mov %r10, 8(%rsp)
6: and $0x00ff, %rbx
7: imul %rax, %rbx
8: lea var_b, %rax
9: mov %rbx, 16(%rsp)

READY = { 3, 5 } ← 8, 9
Example

1: lea    var_a, %rax
2: add    4(%rsp), %rax
3: inc    %r11
4: mov    4(%rsp), %r10
5: mov    %r10, 8(%rsp)
6: and    $0x00ff, %rbx
7: imul   %rax, %rbx
8: lea    var_b, %rax
9: mov    %rbx, 16(%rsp)

READY = \{ 3, 5, 8, 9 \}

<table>
<thead>
<tr>
<th>ALUop</th>
<th>1</th>
<th>6</th>
<th>2</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEM 1</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MEM 2</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example

1: lea var_a, %rax
2: add 4(%rsp), %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: mov %r10, 8(%rsp)
6: and $0x00ff, %rbx
7: imul %rax, %rbx
8: lea var_b, %rax
9: mov %rbx, 16(%rsp)

READY = { 3, 5, 8, 9 }

<table>
<thead>
<tr>
<th>ALUop</th>
<th>1</th>
<th>6</th>
<th>2</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEM 1</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MEM 2</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example

1: lea var_a, %rax
2: add 4(%.rsp), %rax
3: inc %r11
4: mov 4(%.rsp), %r10
5: mov %r10, 8(%.rsp)
6: and $0x00ff, %rbx
7: imul %rax, %rbx
8: lea var_b, %rax
9: mov %rbx, 16(%.rsp)

READY = { 3, 5, 8, 9 }
Example

1: lea var_a, %rax
2: add 4(%rsp), %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: mov %r10, 8(%rsp)
6: and $0x00ff, %rbx
7: imul %rax, %rbx
8: lea var_b, %rax
9: mov %rbx, 16(%rsp)

\[ \text{READY} = \{5, 8, 9\} \]

<table>
<thead>
<tr>
<th>ALUop</th>
<th>1</th>
<th>6</th>
<th>3</th>
<th>2</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEM 1</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MEM 2</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ d=0 \quad f=0 \quad d=2 \quad f=0 \quad d=5 \quad f=1 \quad d=4 \quad f=1 \quad d=0 \quad f=0 \quad d=2 \quad f=1 \quad d=0 \quad f=0 \quad d=1 \quad f=2 \]
Example

1: lea var_a, %rax
2: add 4(%rsp), %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: mov %r10, 8(%rsp)
6: and $0x00ff, %rbx
7: imul %rax, %rbx
8: lea var_b, %rax
9: mov %rbx, 16(%rsp)

READY = { 5, 8, 9 }

<table>
<thead>
<tr>
<th>ALUop</th>
<th>1</th>
<th>6</th>
<th>3</th>
<th>2</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEM 1</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MEM 2</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example

1: lea var_a, %rax  
2: add 4(%rsp), %rax  
3: inc %r11  
4: mov 4(%rsp), %r10  
5: mov %r10, 8(%rsp)  
6: and $0x00ff, %rbx  
7: imul %rax, %rbx  
8: lea var_b, %rax  
9: mov %rbx, 16(%rsp)

READY = { 5, 8, 9 }

<table>
<thead>
<tr>
<th>ALUop</th>
<th>1</th>
<th>6</th>
<th>3</th>
<th>2</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEM 1</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MEM 2</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example

1: lea var_a, %rax
2: add 4(%rsp), %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: mov %r10, 8(%rsp)
6: and $0x00ff, %rbx
7: imul %rax, %rbx
8: lea var_b, %rax
9: mov %rbx, 16(%rsp)

READY = { 8, 9 }

<table>
<thead>
<tr>
<th>ALUop</th>
<th>1</th>
<th>6</th>
<th>3</th>
<th>2</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEM 1</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MEM 2</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example

1: lea var_a, %rax
2: add 4(%rsp), %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: mov %r10, 8(%rsp)
6: and $0x00ff, %rbx
7: imul %rax, %rbx
8: lea var_b, %rax
9: mov %rbx, 16(%rsp)

READY = { 8, 9 }

<table>
<thead>
<tr>
<th>ALUop</th>
<th>1</th>
<th>6</th>
<th>3</th>
<th>2</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEM 1</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MEM 2</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example

1: lea var_a, %rax
2: add 4(%rsp), %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: mov %r10, 8(%rsp)
6: and $0x00ff, %rbx
7: imul %rax, %rbx
8: lea var_b, %rax
9: mov %rbx, 16(%rsp)

READY = { 8, 9 }
Example

1: lea var_a, %rax
2: add 4(%rsp), %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: mov %r10, 8(%rsp)
6: and $0x00ff, %rbx
7: imul %rax, %rbx
8: lea var_b, %rax
9: mov %rbx, 16(%rsp)

READY = { 9 }
Example

1: lea var_a, %rax
2: add 4(%rsp), %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: mov %r10, 8(%rsp)
6: and $0x00ff, %rbx
7: imul %rax, %rbx
8: lea var_b, %rax
9: mov %rbx, 16(%rsp)

READY = { 9 }

<table>
<thead>
<tr>
<th>ALUop</th>
<th>1</th>
<th>6</th>
<th>3</th>
<th>2</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEM 1</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MEM 2</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example

1: lea var_a, %rax
2: add 4(%rsp), %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: mov %r10, 8(%rsp)
6: and $0x00ff, %rbx
7: imul %rax, %rbx
8: lea var_b, %rax
9: mov %rbx, 16(%rsp)

READY = { 9 }

<table>
<thead>
<tr>
<th>ALUop</th>
<th>1</th>
<th>6</th>
<th>3</th>
<th>2</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEM 1</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
<td>9</td>
</tr>
<tr>
<td>MEM 2</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example

1: lea var_a, %rax
2: add 4(%rsp), %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: mov %r10, 8(%rsp)
6: and $0x00ff, %rbx
7: imul %rax, %rbx
8: lea var_b, %rax
9: mov %rbx, 16(%rsp)

READY = { }
Example

1: lea var_a, %rax
2: add 4(%rsp), %rax
3: inc %r11
4: mov 4(%rsp), %r10
5: mov %r10, 8(%rsp)
6: and $0x00ff, %rbx
7: imul %rax, %rbx
8: lea var_b, %rax
9: mov %rbx, 16(%rsp)

READY = { }
Outline

• Modern architectures
• Introduction to instruction scheduling
• List scheduling
• Resource constraints
• Scheduling across basic blocks
• Trace scheduling
Scheduling across basic blocks

• Number of instructions in a basic block is small
  – Cannot keep a multiple units with long pipelines busy by just scheduling within a basic block

• Need to handle control dependence
  – Scheduling constraints across basic blocks
  – Scheduling policy
Moving across basic blocks

- Downward to adjacent basic block
Moving across basic blocks

- Downward to adjacent basic block
Moving across basic blocks

- Downward to adjacent basic block

- A path to B that does not execute A?
Moving across basic blocks

• Upward to adjacent basic block
Moving across basic blocks

• Upward to adjacent basic block
Moving across basic blocks

- Upward to adjacent basic block

- A path from C that does not reach A?
Control Dependencies

- Constraints in moving instructions across basic blocks
Control Dependencies

- Constraints in moving instructions across basic blocks

$$\text{if ( \ldots )}$$
$$a = b \ \text{op} \ c$$
Control Dependencies

• Constraints in moving instructions across basic blocks

```c
if ( . . . )
a = b \text{ op } c
```
Control Dependencies

- Constraints in moving instructions across basic blocks

```plaintext
if ( c != 0 )
a = b / c
```

**NO!!!**
Control Dependencies

- Constraints in moving instructions across basic blocks

\[
\begin{align*}
\text{If ( . . . . )} \\
d &= *(a1)
\end{align*}
\]
Control Dependencies

• Constraints in moving instructions across basic blocks

If ( valid address? )

d = *(a1)
Outline

• Modern architectures
• Introduction to instruction scheduling
• List scheduling
• Resource constraints
• Scheduling across basic blocks
• Trace scheduling
Trace Scheduling

- Find the most common trace of basic blocks
  - Use profile information

- Combine the basic blocks in the trace and schedule them as one block

- Create clean-up code if the execution goes off-trace
Trace Scheduling
Trace Scheduling

A

B

C

D

E

F

G

H
Trace Scheduling
Trace Scheduling

A

B

D

E

G

H
Trace Scheduling
Large Basic Blocks via Code Duplication

- Creating large extended basic blocks by duplication
- Schedule the larger blocks
Large Basic Blocks via Code Duplication

- Creating large extended basic blocks by duplication
- Schedule the larger blocks
Trace Scheduling

A → B → D → E → G → H
Trace Scheduling