DYNAMIC SCHEDULING

Mahdi Nazm Bojnordi

Assistant Professor
School of Computing
University of Utah
Overview

 Announcement

 Homework 3 will be uploaded tonight (11:59PM)

 This lecture

 Recap Branch Prediction
 Dynamic scheduling
   Forming data flow graph on the fly
 Register renaming
   Removing false data dependence
   Architectural vs. physical registers
Recall: Branch Predictors

1-bit predictor
2-bit predictor
GHR: Correlated
Global Predictor
Local Predictor
Tournament Predictor

DHT: Limited PC Based

BHT: Tagged

BHT + DHT
Recall: Branch Predictors

1-bit predictor

2-bit predictor

GHR: Correlated

DHT: Limited PC Based

PC Based

BHT: Tagged

Local Predictor

BHT + DHT

Global Predictor

Tournament Predictor

Global Predictor
Branch Prediction Summary

- Dedicated predictor per branch
  - Program counter is used for assigning predictors to branches
- Capturing correlation among branches
  - Shift register is used to track history
- Predicting branch direction is not enough
  - Which instruction to be fetched if taken?
- Storing the target instruction can eliminate fetching
  - Extra hardware is required
Branch Target Buffer

- Store a target address for each branch
Branch Target Buffer

- Store tags and target addresses for each branch

```
PC

V  Tag  Target

AND

Target address
Hit/miss*
```
**Goal:** exploiting more ILP by avoiding stall cycles

- Branch prediction can avoid the stall cycles in the frontend
Big Picture

- **Goal**: exploiting more ILP by avoiding stall cycles
  - Branch prediction can avoid the stall cycles in the frontend
  - More instructions are sent to the pipeline
Goal: exploiting more ILP by avoiding stall cycles

- Branch prediction can avoid the stall cycles in the frontend
  - More instructions are sent to the pipeline
- Instruction scheduling can remove unnecessary stall cycles in the execution/memory stage
  - Static scheduling
    - Complex software (compiler)
    - Unable to resolve all data hazards (no access to runtime details)
  - Dynamic scheduling
    - Completely done in hardware
Dynamic Scheduling

- **Key idea**: creating an instruction schedule based on runtime information
  - Hardware managed instruction reordering

Assembly code:

```
DIV  F1, F2, F3
ADD  F4, F1, F5
SUB  F6, F5, F7
```
Dynamic Scheduling

- **Key idea:** creating an instruction schedule based on runtime information
  - Hardware managed instruction reordering

Assembly code:

DIV F1, F2, F3  
ADD F4, F1, F5  
SUB F6, F5, F7

Long latency operation

Dependent instruction
Dynamic Scheduling

- **Key idea:** creating an instruction schedule based on runtime information
  - Hardware managed instruction reordering

Assembly code:

- DIV   F1, F2, F3
- ADD F4, F1, F5
- SUB F6, F5, F7

- Long latency operation
- Dependent instruction
- Independent instruction

Out-of-order execution?
Dynamic Scheduling

- **Key idea:** creating an instruction schedule based on runtime information
  - Hardware managed instruction reordering
  - Instructions are executed in data flow order

Program code

```
ADDI  R1, R0, #1
ADDI  R2, R0, #4
ADD   R3, R3, R2
ADDI  R2, R2, #\text{-}1
BNEQ R2, R1, next
ADD   R4, R4, R3
BNEQ R2, R0, loop
```
Dynamic Scheduling

- **Key idea**: creating an instruction schedule based on runtime information
  - Hardware managed instruction reordering
  - Instructions are executed in data flow order

Program code

```
ADDI R1, R0, #1
ADDI R2, R0, #4
ADD   R3, R3, R2
ADDI  R2, R2, #1
BNEQ R2, R1, next
ADD   R4, R4, R3
BNEQ R2, R0, loop
next:
```
Dynamic Scheduling

- **Key idea:** creating an instruction schedule based on runtime information
  - Hardware managed instruction reordering
  - Instructions are executed in data flow order

<table>
<thead>
<tr>
<th>Program code</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDI R1, R0, #1</td>
</tr>
<tr>
<td>ADDI R2, R0, #4</td>
</tr>
<tr>
<td>ADD R3, R3, R2</td>
</tr>
<tr>
<td>ADDI R2, R2, #-1</td>
</tr>
<tr>
<td>BNEQ R2, R1, next</td>
</tr>
<tr>
<td>BNEQ R2, R0, loop</td>
</tr>
<tr>
<td>ADD R3, R3, R2</td>
</tr>
<tr>
<td>ADDI R2, R2, #-1</td>
</tr>
<tr>
<td>BNEQ R2, R1, next</td>
</tr>
<tr>
<td>BNEQ R2, R0, loop</td>
</tr>
<tr>
<td>ADD R3, R3, R2</td>
</tr>
<tr>
<td>ADDI R2, R2, #-1</td>
</tr>
<tr>
<td>BNEQ R2, R1, next</td>
</tr>
<tr>
<td>BNEQ R2, R0, loop</td>
</tr>
<tr>
<td>ADD R4, R4, R3</td>
</tr>
<tr>
<td>BNEQ R2, R0, loop</td>
</tr>
<tr>
<td>ADD R3, R3, R2</td>
</tr>
<tr>
<td>ADDI R2, R2, #-1</td>
</tr>
<tr>
<td>BNEQ R2, R1, next</td>
</tr>
<tr>
<td>BNEQ R2, R0, loop</td>
</tr>
<tr>
<td>ADD R4, R4, R3</td>
</tr>
<tr>
<td>BNEQ R2, R0, loop</td>
</tr>
<tr>
<td>ADD R3, R3, R2</td>
</tr>
<tr>
<td>ADDI R2, R2, #-1</td>
</tr>
<tr>
<td>BNEQ R2, R1, next</td>
</tr>
<tr>
<td>BNEQ R2, R0, loop</td>
</tr>
</tbody>
</table>

- next:

- loop:
Dynamic Scheduling

- **Key idea:** creating an instruction schedule based on runtime information
- Hardware managed instruction reordering
- Instructions are executed in data flow order
Dynamic Scheduling

- **Key idea:** creating an instruction schedule based on runtime information
  - Hardware managed instruction reordering
  - Instructions are executed in data flow order

How to form data flow graph on the fly?
Register Renaming

- Eliminating WAR and WAW hazards
  - Change the mapping between architectural registers and physical storage locations

DIV F1, F2, F3
ADD F4, F1, F5
SUB F5, F6, F7
ADD F4, F5, F8
Register Renaming

- Eliminating WAR and WAW hazards
  - Change the mapping between architectural registers and physical storage locations

RAW

DIV  F1, F2, F3
ADD  F4, F1, F5
SUB  F5, F6, F7
ADD  F4, F5, F8
Register Renaming

- Eliminating WAR and WAW hazards
  - Change the mapping between architectural registers and physical storage locations
Register Renaming

- Eliminating WAR and WAW hazards
  - Change the mapping between architectural registers and physical storage locations
Register Renaming

- Eliminating WAR and WAW hazards
  - Change the mapping between architectural registers and physical storage locations
Register Renaming

- Eliminating WAR and WAW hazards
  - Change the mapping between architectural registers and physical storage locations

WAR and WAW hazards can be removed using more registers
Eliminating WAR and WAW hazards

1. allocate a free physical location for the new register
2. find the most recently allocated location for the register

DIV F1, F2, F3
ADD F4, F1, F5
SUB F5, F6, F7
ADD F4, F5, F8
Register Renaming

- Eliminating WAR and WAW hazards
  - 1. allocate a free physical location for the new register
  - 2. find the most recently allocated location for the register

DIV F1, F2, F3
ADD F4, F1, F5
SUB F5, F6, F7
ADD F4, F5, F8

DIV P12, P11, P10
Register Renaming

- Eliminating WAR and WAW hazards
  - 1. allocate a free physical location for the new register
  - 2. find the most recently allocated location for the register

<table>
<thead>
<tr>
<th>architectural Registers</th>
<th>physical Locations</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>P10</td>
</tr>
<tr>
<td>F2</td>
<td>P11</td>
</tr>
<tr>
<td>F3</td>
<td>P12</td>
</tr>
<tr>
<td>F4</td>
<td>P13</td>
</tr>
<tr>
<td>F5</td>
<td>P14</td>
</tr>
<tr>
<td>F6</td>
<td>P15</td>
</tr>
<tr>
<td>F7</td>
<td>P16</td>
</tr>
<tr>
<td>F8</td>
<td>P17</td>
</tr>
<tr>
<td>P10, P11, P12</td>
<td>P18</td>
</tr>
<tr>
<td>P13, P14</td>
<td>P19</td>
</tr>
</tbody>
</table>

- DIV F1, F2, F3
- ADD F4, F1, F5
- SUB F5, F6, F7
- ADD F4, F5, F8
- DIV P12, P11, P10
- ADD P14, P12, P15
Eliminating WAR and WAW hazards

1. allocate a free physical location for the new register
2. find the most recently allocated location for the register
Register Renaming

- Eliminating WAR and WAW hazards
  - 1. allocate a free physical location for the new register
  - 2. find the most recently allocated location for the register

<table>
<thead>
<tr>
<th>Architectural Registers</th>
<th>Physical Locations</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>P10</td>
</tr>
<tr>
<td>F2</td>
<td>P11</td>
</tr>
<tr>
<td>F3</td>
<td>P12</td>
</tr>
<tr>
<td>F4</td>
<td>P13</td>
</tr>
<tr>
<td>F5</td>
<td>P14</td>
</tr>
<tr>
<td>F6</td>
<td>P15</td>
</tr>
<tr>
<td>F7</td>
<td>P16</td>
</tr>
<tr>
<td>F8</td>
<td>P17</td>
</tr>
</tbody>
</table>

- DIV F1, F2, F3
- ADD F4, F1, F5
- SUB F5, F6, F7
- ADD F4, F5, F8

- DIV P12, P11, P10
- ADD P14, P12, P15
- SUB P19, P17, P13
- ADD P18, P19, P16