Pipelining
Performance improvement of single cycle processor implementation discussed in chapter 5

Chapter 4
Performance improvement of ALU components
Adders
Multipliers
Dividers
Floating point adders
Considered
Datapath
Control path

Chapter 6
Datapath
Control path
For single cycle pipelined processor

Pipelining will improve performance if there are
No data dependencies between two instructions
No branch instructions
No exceptions
Problems with pipelining
Hazards
Data
Branch
Data Hazards
Solutions
Data hazard and forwarding
Data hazard and stalls
Branch hazards

Exceptions

Superscalar and dynamic pipelining

Pipelining
Improves performance
By increasing instruction throughput
Not by decreasing execution time of an individual instruction as we learned in chapter 2
In chapter performance = 1/CPU time
No change in execution time per instruction
In general
Speed up = number of stages in pipeline

Complications
Hazards
Structural
Control
Data
Solutions
Forwarding
Prediction
Pipelined data path
  Divide the instruction execution into 5 stages
  5 stage pipelining for MIPS instruction set
  Introduce pipeline registers to store intermediate results in between stages

5 stages for instruction execution

  instruction fetch IF

  Instruction decode/Register Fetch  ID

  Instruction execution  EX

  Memory access  MEM

  Write back  WB

IF  PC

  INST Memory

ID  Register file

EX  ALU

MEM  Data Memory

WB  Register file
Example

Represent the 5 stage pipeline execution of
Lw $10 20($1)
Sub $11 $2 $3

Pipeline datapath correction
Initially write register number is supplied by the IF/ID pipeline register
However write occurs at the end
Preserve the number in ID/EX, EX/MEM, MEM/WB pipeline registers

Example

Represent the 5 stage pipeline execution of
Lw $10 20($1)
Sub $11 $2 $3
Total execution time with pipeline 6 ns

Total execution time with no pipeline registers 5 X 2 ns

Pipeline control path
   Generate control values and pass them down the pipeline with data

Data hazard problem and forwarding solution
   Instruction with data dependency will not execute properly unless the data hazard is resolved

   Data hazard can be resolved by detecting dependency and forwarding data directly from appropriate pipeline register

Example
   Illustrate the forwarding involved with the execution of following instructions

   Add $0 $1 $2
   Sub  $3 $0 $4

   Forwards data from EX stage of add instruction to the input of EX stage for sub
   Replacing the value from register 0 read in the second stage of Sub

Data hazard and stalls
   Forwarding cannot solve the problem when an instruction tries to read a register following a load instruction
Hazard detection unit capable of stalling the pipeline is needed to properly handle this hazard

Example
Illustrate the forwarding path involved with the execution of following instructions graphically

\[
\begin{align*}
\text{Lw} & \ $2 \ 20($1) \\
\text{Sub} & \ $4 \ $2 \ $3
\end{align*}
\]

Shows the data dependency related to memory

Use shades to show reads/writes
- Right half shade - read
- Left half shade - write

Example
Show the pipelining with the execution of following instructions

\[
\begin{align*}
\text{sub} & \ $2, \ $1, \ $3 \\
\text{and} & \ $12, \ $2, \ $5 \\
\text{or} & \ $13, \ $6, \ $2 \\
\text{add} & \ $14, \ $2, \ $2 \\
\text{sw} & \ $15, \ 100($2)
\end{align*}
\]
Branch hazard
When a branch instruction completes there may already be instructions in the pipeline

If wrong instruction in the pipeline - flush

Predict
Predict that branch will fail
If right pipeline at full speed
If branch succeeds - stall

Delayed branch
Replace the pipe stall (bubble) with an un affected instruction

Add $4 $5 $6
Beq $1 $2 40
Lw $3 300($0)

TO
Beq $1 $2 40
Add $4 $5 $6
Lw $3 300($0)

Example
Reorder the following code to avoid pipeline stalls
Lw $t0 0($t1)
Lw $t2 4 ($t1)
Sw $t2 0($t1)
Sw $t0 4($t1)

Solution
Lw $t0 0($t1)
Lw $t2 4 ($t1)
Sw $t0 4($t1)
Sw $t2 0($t1)
Another way to avoid stalls
   Follow a load with an instruction independent of that load

Size of pipeline registers
   IF/ID PL - 64 bits = (pc+4) next inst 32 + current inst 32
   
   ID/EX PL - 128 bits = sign ext 32 + read data from reg2 + read data from reg 1 + next instruction
   
   EX/MEM - 97 bits = data from reg 2 (32) + ALU result (32) + 1 zero bit + next instruction (32)
   
   MEM/WB - 64 = address 32 + read data (32)

Example
   Shade the portions of data path for lw instruction
   Refer to page 454 455 456

Example
   Shade the portions of data path for sw instruction
   Refer to page 457 459

Example
   Draw the pipeline diagrams for two instructions
   Lw $10 20($1)
   Sub $11 $2 $3
   
   Refer to page 463 464 465
Lecture 27 Pipelined control path
Single cycle implementation
PC written every clock cycle
No separate write signal for PC
Pipeline registers written during each cycle
No write signals for pipeline registers

EX/Address calculation stage 4 control signals
- regdst, aluop1, aluop0, alusrc

memory access stage 3 control signals
- branch, memread, memwrite

write back stage 2 control signals
- regwrite, memto reg

<table>
<thead>
<tr>
<th>Inputs</th>
<th>outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>op5</td>
<td>op4</td>
</tr>
<tr>
<td>op3</td>
<td>op2</td>
</tr>
<tr>
<td>op1</td>
<td>op0</td>
</tr>
<tr>
<td>RegDst</td>
<td>Alu</td>
</tr>
<tr>
<td>OP1</td>
<td>AluOP2</td>
</tr>
<tr>
<td>Alusrc</td>
<td>branch</td>
</tr>
<tr>
<td>MemRead</td>
<td>MemWrite</td>
</tr>
<tr>
<td>RegWrite</td>
<td>memtoReg</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Execution/Address Calculation stage control lines</th>
<th>Memory access stage control lines</th>
<th>Write-back stage control lines</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reg Dst</td>
<td>ALU</td>
<td>ALU</td>
</tr>
<tr>
<td>R-format</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>lw</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>sw</td>
<td>X</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>beq</td>
<td>X</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Pipelined control path

- Single cycle implementation
  o PC written every clock cycle
  o No separate write signal for PC
- Pipeline registers written during each cycle
  o No write signals for pipeline registers

4 control signals
  Ex/ Address Calculation Stage: Reg Dst, ALU op1, ALU op0, ALU Src

3 control signals
  Memory Access Stage: Branch, Mem Read, Mem Write

2 control signals
  Write Back Stage: Reg Write, Mem To Reg
Identify the control signals generated with lw
0001 010 11

Identify the control signals generated with R-format
1100 000 10

Identify the control signals generated with sw
X001 001 1x

Identify the control signals generated with beq
X010 100 0x
Control signals with instructions

Ex/ Address Calculation   Mem. Access   Write Back
Reg Dst | ALU op1 | ALU op2 | ALU src | Branch | Mem | Mem | Reg | Mem to | Write | Write | Reg |
| Read | Write |                      Write |  Reg      |

Ex: Control signal generated with lw

Ex: Control signal generated with R-format

Ex: Control signal generated with sw

Ex: Control signal generated with beq
Ex: Show the control signals generated for the following 5 instructions when going through the pipeline
Lw
Sub
And
Or
Add

Stage 1:

Lw

Stage 2:

Sub

Stage 3:

And

See pages 471
Figures 6.31 – 6.35
Ex: Show the control signals generated for the following 2 instructions program
Lw
Sub
Stage 1:

Stage 2:

Stage 3:
Stage 4: no instruction

Controller

ID / EX

EX | MEM | WB

0 0 0 0 0 0 0 0

EX / MEM from sub

MEM | WB

0 0 0 100

MEM / WB from lw

1 1 0 0
Control signals

Rs  Rt  Rt  Rd
Mem / WB .RegisterRead
Ex / Mem .RegisterRead
Ex / Mem .RegisterWrite

FORWARDING UNIT

Forward A:
00 \(\rightarrow\) Source ID / EX
10 \(\rightarrow\) Source EX / MEM
01 \(\rightarrow\) Source MEM / WB

Forward B:
00 \(\rightarrow\) ID / EX
10 \(\rightarrow\) EX / MEM
01 \(\rightarrow\) MEM / WB
Pipeline Data Path Correction

IF / ID

WD  WR  RR2  RR1
R-File

ID / EX

EX / MEM

MEM / WB

MUX
Initially write register number is supplied by the IF / ID pipeline register

However, write occurs at the end.

Preserve the number in ID / EX

EX / MEM

MEM / WB

pipeline register

Example

Identify the control signals generated for two instructions lw and sub for each stage of the pipeline

Example

Identify the control signals generated for two instructions lw, sub, and, or, add for each stage of the pipeline

Refer pages 471-476

Hazard detection

EX hazard

If (EX/Mem.regwrite
And (EX/Mem.RegRd =/=0)
And (Ex/Mem.RegRd = ID/EX. RegisterRs))
Then
ForwardA = 10

If (EX/Mem.regwrite
And (EX/Mem.RegRd =/=0)
And (Ex/Mem.RegRd = ID/EX. RegisterRt))
Then
ForwardB = 10
Mem Hazard
If (mem/wb.Regwrite
And (mem/wb.registerRd /= 0)
And (mem/wb.registerRd = ID/Ex.registe rs))
Then
ForwardA = 01

If(mem/WB.regwrite
And(mem/wb.registerRd=/=0
And (mem/wb.registerRd = ID/Ex.registerRt))
Then
ForwardB = 01