Data Hazards

Philipp Koehn

28 March 2018
Data Hazard

• Definition: instruction waits on result from prior instruction

• Example
  
  add $s0, $t0, $t1
  sub $t0, $s0, $t3

  – add instruction writes result to register $s0 in stage 5
  – sub instruction reads $s0 in stage 2

⇒ Stage 2 of sub has to be delayed

• We overcome this in hardware
Graphical Representation

add $s0,$t0,$t1

• IF: instruction fetch
• ID: instruction decode
• EX: execution
• MEM: memory access
• WB: write-back
Add and Subtract

- Add wiring to circuit to directly connect output of ALU for next instruction
Load and Subtract

lw $s0,20($t0)

EX

MEM

wb

sub $t0,$s0,$t3

EX

MEM

• Add wiring from memory lookup to ALU

• Still 1 cycle unused: "pipeline stall" or "bubble"
forwarding
Add and Subtract

- Example

  add $s0, $t0, $t1
  sub $t0, $s0, $t3

- Plan
Add (Stage 1)

IF: Instruction Fetch

ID: Instruction decoder
    register file read

EX: Execute /
    address calculate

MEM: Memory access

WB: Write Back

Selector
PC

Add

Instruction Memory
Instruction

Address

Register 1

Register 2

Registers

Read register 1
Read register 2
Write register
Write data

Read data 1
Read data 2

Write data

Add

Sign extended

Shift Left

Zero

ADD

ALU

Result

Selector

Address

Read data

Data Memory

Write data
Subtract (Stage 1), Add (Stage 2)
Subtract (Stage 2), Add (Stage 3)

IF: Instruction Fetch
ID: Instruction decoder register file read
EX: Execute / address calculate
MEM: Memory access
WB: Write Back

Selector
PC

Selector

Add

4

Address
Instruction Memory
Instruction

Read register 1
Read register 2
Write register
Write data

Read data 1
Read data 2

Sign extended
Shift Left

ADD

Zero

ALU
Result

Selector

Address
Data Memory
Write data

Philipp Koehn
Computer Systems Fundamentals: Data Hazards
28 March 2018
Subtract (Stage 3), Add (Stage 4)
Data Hazard
Forwarding Data

IF: Instruction Fetch

ID: Instruction decoder
  register file read

EX: Execute /
  address calculate

MEM: Memory access

WB: Write Back

Selector
PC

Address

Instruction Memory

Instruction

Add

4

Read register 1
Read register 2
Write register
Write data

Selector

Data

Memory

ALU

Add

Selector

Sign extended

Shift Left

ADD

Selector

Zero

ALU Result

Selector

Address
Read data
Data Memory
Write data

Write data
Forwarding Unit

- Forwarding Unit must
  - detect if there is a data hazard
  - forward the right register values

- Relevant information for decision
  - identify of input registers used in instruction currently in EX (either first or second operand)
  - identity of output register used in instruction currently in MEM
  - value of output register used in instruction currently in MEM

- Format of decision
  - Register value
  - Control lines for selectors for input to ALU
Formal Names

• Relevant information for decision
  – **EX.Rs** and **EX.Rt**
    identify of input registers used in instruction currently in EX (either first or second operand)
  – **MEM.Rd**
    identity of output register used in instruction currently in MEM
  – **MEM.RdValue**
    value of output register used in instruction currently in MEM

• Format of decision
  – **Forward.Rs** and **Forward.Rt**
    Register value
  – **Hazard.Rs** and **Hazard.Rt**
    Control lines for selectors for input to ALU
Forwarding Logic

• Logic in forwarding unit

    if (MEM.Rd == EX.Rs)
        Forward.Rs = MEM.RdValue
        Hazard.Rs = 1
    else
        Hazard.Rs = 0

    if (MEM.Rd == EX.Rt)
        Forward.Rt = MEM.RdValue
        Hazard.Rt = 1
    else
        Hazard.Rt = 0

• Must also check if "RegisterWrite" for instruction in MEM stage

• Relevant information must be passed through stages
stalling
Load and Subtract

• Example

\[\text{load } s0, 20(t0)\]
\[\text{sub } t0, s0, t3\]

• Plan

```
IF  ID  EX  MEM  WB
200 400 600 800 1000 1200
lw \$s0, 20($t0)

IF  ID  EX  MEM  WB

sub \$t0, \$s0, \$t3
```

bubble bubble bubble bubble bubble
Load (Stage 1)

IF: Instruction Fetch

ID: Instruction decoder
- register file read

EX: Execute /
- address calculate

MEM: Memory access

WB: Write Back

Selector
PC
Address
Instruction Memory
Instruction

Add

4

Sign extended

Shift Left

ADD

Zero

ALU Result

Address
Data Memory
Write data

Selector

Read register 1
Read data 1

Read register 2
Read data 2

Write register
Write data

Selector

Read
Write
Subtract (Stage 1), Load (Stage 2)
Subtract (Stage 2), Load (Stage 3)
Subtract (Stage 3), Load (Stage 4)
Data Hazard

IF: Instruction Fetch

ID: Instruction decoder
register file read

EX: Execute / address calculate

MEM: Memory access

WB: Write Back

Selector

PC

Add

Sign extended

Shift Left

Zero

ALU Result

Selector

Address

Instruction memory

Instruction

Register file read

MEM: Memory access

Write data

Write register

Read data 1

Read register 1

Read data 2

Read register 2

Selector

Write data

Write register

Selector

Data Memory

Read data

Address

Selector

4

ADD

Zero
Data Hazard

• Our example

\[
\text{load } s0, 20(t0) \\
\text{sub } t0, s0, t3
\]

• Worse than add/sub hazard
  – we need operand value in $s0$
  – we have not even retrieved it at this stage

• Stalling
  – load instruction processing has to move to stage 5
  – sub instruction processing has to stall
Stalling

- Hazard condition between 2 instructions
- Second instruction has to be delayed
- Technical solution: insert a "nop" operation ("no operation")
- Resets program counter
• Fetch of load instruction
Load and Sub Processing

- Fetch of sub instruction

```
IF
lw $s0,20($t0)
ID
IF
sub $t0,$s0,$t3
```
Next stage

- load: address calculation
- sub: instruction decode
Load and Sub Processing

lw $s0,20($t0)

sub $t0,$s0,$t3

• Next stage
  - load: address calculation
  - sub: instruction decode

• Registers are known now $\rightarrow$ hazard detected
Load and Sub Processing

- Insertion of nop instruction

```
lw $s0,20($t0)
sub $t0,$s0,$t3
```
Load and Sub Processing

- Next stage
  - load: memory retrieve
  - sub: instruction decode
Load and Sub Processing

- Next stage
  - load: write to register
  - sub: ALU operation execution

- Operand for sub forwarded from load instruction execution
Hazard Detection (Stalling) Unit

- Stalling unit must
  - detect if there is a data hazard
  - insert a "nop" instruction into pipeline

- Relevant information for decision
  - identify of input registers used in instruction currently in ID (either first or second operand)
  - identity of load register used in instruction currently in EX
  - control flag that there is indeed a memory read in EX

- Format of decision
  - overwrite instruction currently in ID with "nop"
  - reset program counter
Formal Names

• Relevant information for decision
  – **ID.Rs** and **ID.Rt**
    identify of input registers used in instruction currently in ID (either first or second operand)
  – **EX.Rd**
    identity of load register used in instruction currently in EX
  – **EX.MemRead**
    control flag that there is indeed a memory read in EX

• Format of decision
  – **ID/EX**
    overwrite instruction currently in ID with "nop"
  – **PC**
    reset program counter
Stalling Logic

- Logic in stalling unit

if (EX.MemRead and
    (EX.Rd = ID.Rs or
     EX.Rd = ID.Rt))
  PC = PC - 4
  ID/EX = nop
Stalling Unit

IF: Instruction Fetch

ID: Instruction decoder register file read

Hazard Detection Unit

Selector

PC

Address

Instruction Memory

Instruction

Registers

Read register 1

Read data 1

Read register 2

Write register

Write data

Read data 2

EX.MemRead

EX.Rd

ID/EX

ID.Rs

ID.Rt

36

Philipp Koehn

Computer Systems Fundamentals: Data Hazards

28 March 2018
Additional Forwarding Logic

- Additional logic in forwarding unit

```java
if (WB.Rd == EX.Rs)
    Forward.Rs = WB.RdValue
    Hazard.Rs = 1
else
    Hazard.Rs = 0

if (WB.Rd == EX.Rt)
    Forward.Rt = WB.RdValue
    Hazard.Rt = 1
else
    Hazard.Rt = 0
```

- Also relevant in "add, anything, add" sequence where result from first add is used in last add
Forwarding Unit

IF: Instruction Fetch
ID: Instruction decoder
MEM: Memory access
WB: Write Back

Selector
PC
Instruction Memory

Forwarding Unit

4
Add

Instruction

Register file read

EX: Execute / address calculate

Zero

ALU

WB:
Write Back

Selector

Address

Read register 1
Read data 1

Read register 2
Read data 2

Read register

Write register

Write data

Selector

Shift Left

Sign extended

Selector

Address

Read data

Data Memory

Write data

Selector

Select

Add

Selector

Select

Selector

Select

Selector

Select