Lecture 5: Dependence Analysis and Superscalar Techniques Overview

Instruction dependences, correctness, inst scheduling examples, renaming, speculation, generic superscalar pipelines

Sequential Execution Model

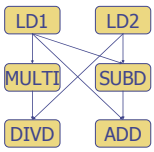
Any program execution is "correct" if the final architectural states (registers and memory contents) is the same as by sequential execution

Single-cycle implementation is intuitively correct

If instructions are not executed sequentially, what is "correct" execution?

Data Flow Execution

```
LD     F2,0(R3)
MULTI  F0,F2,F4
LD     F6,0(R2)
SUBD   F8,F6,F2
DIVD   F10,F0,F6
ADD    F12,F8,F2
```



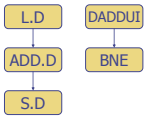
Note: no branch in this code

Data Dependences

- Instruction J is dependent on I if
  - I's output is used by J, or
  - J is dependent on K, and K is dependent on I

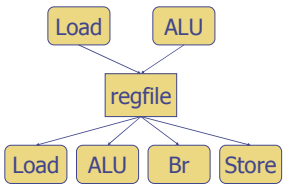
```
Loop: L.D    F0,0(R1)
      ADD.D  F4,F0,F2
      S.D    F4,0(R1)
      DADDUI R1,R1,#-8
      BNE    R1,R2,LOOP
```

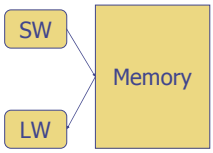
- Data Dependence Graph



Data Dependences

- Dependences through registers
- Dependence through memory






```
ADD r8, r9, r10
BEQ r8, r11, loop
```

```
SW r8, 100(r9)
LW r10, 100(r9)
```


Name Dependences

- Antidependence (WAR): one instruction overwrite a register or memory location that a prior instruction reads

```
LW R1, 100(R2)
ADD R2, R3, R4
```

- Output dependence (WAW): two instructions write the same register or memory location

```
LW R1, 100(R2)
Add R2, R1, R2
Add R1, R3, R4
```

Those dependences can be removed

Dependences vs Hazards

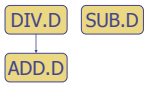
- Dependences are properties of programs
- Hazards are properties of pipelines
- Dependences indicates the **potential** of hazards
- Pipeline implementations determine actual hazards and the length of any stall

What hazards are exposed by MIPS 5-stage pipeline?

Dynamic Scheduling

General idea: when an instruction stalls, look for independent instructions following it

```
DIV.D F0, F2, F4
ADD.D F10, F0, F8
SUB.D F12, F8, F14
```



- Instruction window: how far to look ahead
- Out-of-order execution
- Respect data dependence

What hazards would be exposed?

Machine Correctness

Let E(M,P) be the execution of P at a given machine M

E(M,P) is correct if and only if E(M,P) = E(S,P): The register and memory contents at the end of E(D,P) are the same as those of a sequence execution.

Machine Correctness

Let E(D,P) be the execution of P on a dynamically scheduled machine D

E(D,P) = E(S,P) if

- E(D,P) and E(S,P) execute the same set of instructions
- For any inst *i*, *i* produces the same output as in E(D,P) and E(S,P)
- Any register or memory word receives the output from the same instruction in E(D,P) and in E(S,P)

Machine Correctness

- For any inst *i*, *i* produces the same output as in E(D,P) and E(S,P)

⇒ For any inst *i*, *i* receives the same inputs in E(D,P) as in E(S,P)

⇒ For any inst *i*, *i* receives the outputs in E(D,P) of its parents in E(S,P)

- Any register or memory work receives the output from the same instruction in E(D,P) and in E(S,P)

⇒ In E(D,P) any register or memory word receives the output of inst *j*, where *j* is the last instruction writes to the register or memory word in E(S,P)

Machine Correctness

E(D,P) = E(S,P) if

- E(D,P) and E(S,P) execute the same set of instructions
- For any inst *i*, *i* receives the outputs in E(D,P) of its parents in E(S,P)
- In E(D,P) any register or memory word receives the output of inst *j*, where *j* is the last instruction writes to the register or memory word in E(S,P)

Data Dependence between Operations

ALU to ALU

```
SUBD   F8,F6,F2
ADD    F6,F8,F2
```

```
SUBD    ADD
IF      IF
ID      ID
EX      --
WB      EX
        WB
```

Load and other insts

```
LD     F2,0(R3)
MULTI  F0,F2,F4
```

```
LD      MULTI
IF      IF
ID      ID
EX      --
MEM     --
WB      EX
        WB
```

13

Dependences between Operations

Store to load //R3+100==R4?

```
S.D   F6,100(R3)
L.D   F2,0(R4)
```

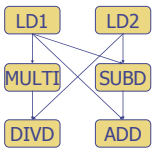
```
S.D     L.D
IF      IF
ID      ID
EX      EX
MEM  ?  --
WB      MEM
```

- Register instruction can be detected by matching register index
- Detecting memory dependence is more difficult

14

Dynamic Scheduling

```
L.D      F2,0(R3)
MULTI    F0,F2,F4
L.D      F6,0(R2)
SUB.D    F8,F6,F2
DIV.D    F10,F0,F6
ADD.D    F12,F8,F2
```



How to schedule pipeline operations?

15

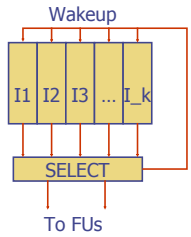
Is This Working?

Inst	IF	ID	Schd	EXE	MEM	WB
L.D	1	2	**3**	4	5	6
MULT	1	2	**3-5**	6-11	-	12
L.D	2	3	**4**	5	6	7
SUB.D	2	3	**4-6**	7-8	9	10
DIV.D	3	4	**5-11**	12-31	11	12
Add.D	3	4	**5-8**	9-10	12	14

Assume (1) two-way issue; (2) FU delay as implied

16

Dynamic Scheduling Implementation



Scoreboarding:
- 1966: scoreboarding in CDC6600

Tomasulo:
- Three years later in IBM 360/91
- Introducing register renaming
- Use tag-based instruction wakeup

17

Name Dependences and Register Renaming

Original code:

```
ADD R3, R1, R2
SUB R4, R4, R3
ADD R3, R6, R7
SUB R3, R3, R4
```

What prevents parallelism?

Renamed code:
R3, R4, R3, R3 renamed to P6, P7, P8, P9 sequentially

```
ADD P6, R1, R2
SUB P7, R4, P6
ADD P8, R6, R7
SUB P9, R5, P7
```

Finally R3 <= P9, R4 <= P7

18

Register Renaming and Correctness

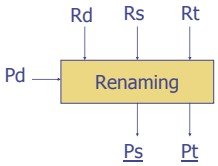
- E(D,P) and E(S,P) execute the same set of instructions
- ⇨ For any inst *i*, *i* receives the outputs in E(D,P) of its parents in E(S,P)
- ⇨ Any register or memory word receives the output of inst *j*, where *j* is the last instruction writes to the register or memory word in E(S,P)

Renaming Implementation

First proposed in Tomasulo (1969)
- Use register status table
- Renamed to reservation station

In P-III
- Use register alias table
- Renamed arch. register to physical register
- Data copied back to arch. register

In other processors (e.g. Alpha 21264, Intel P4)
- Use register mapping table
- No separate architectural/physical registers; no copy-back



Speculative Execution

Modern processors must speculate!
- Branch prediction: SPEC2k INT has one branch per seven instructions!
- Precise interrupt
- Memory disambiguation
- More performance-oriented speculations

Two disjointed but connected issues:
1. **How to make the best prediction**
2. **What to do when the speculation is wrong**

Speculative Execution

Previous correctness condition: E(D,P) and E(S,P) executes the same set of instructions, and …

Now:
- E(Sp, P) commits the same set of instructions as E(S, P) executes
- For any committed inst *i* in E(Sp, P), *i* receives the outputs in E(Sp,P) of its parents in E(S,P)
- In E(Sp, P) any register or memory word receives the output of a committed inst *j*, where *j* is the last inst that writes to the register or memory word in E(Sp, P)

Control Speculation

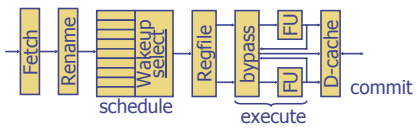
Branch prediction – control speculation
- Must predict on branches
- What to predict
- Branch direction
- Branch target address

What info can be used
- PC value
- Previous branch outputs
- also use branch pattern in complex branch predictors

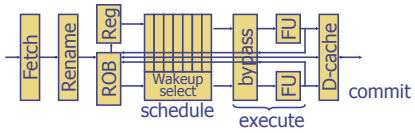
What building blocks are need
- Branch prediction table (BHT), branch target buffer (BTB), pattern registers, and some logics

Generic Superscalar Processor Models

Issue queue based



Reservation based



Revised from Paracharla PhD thesis 1998