

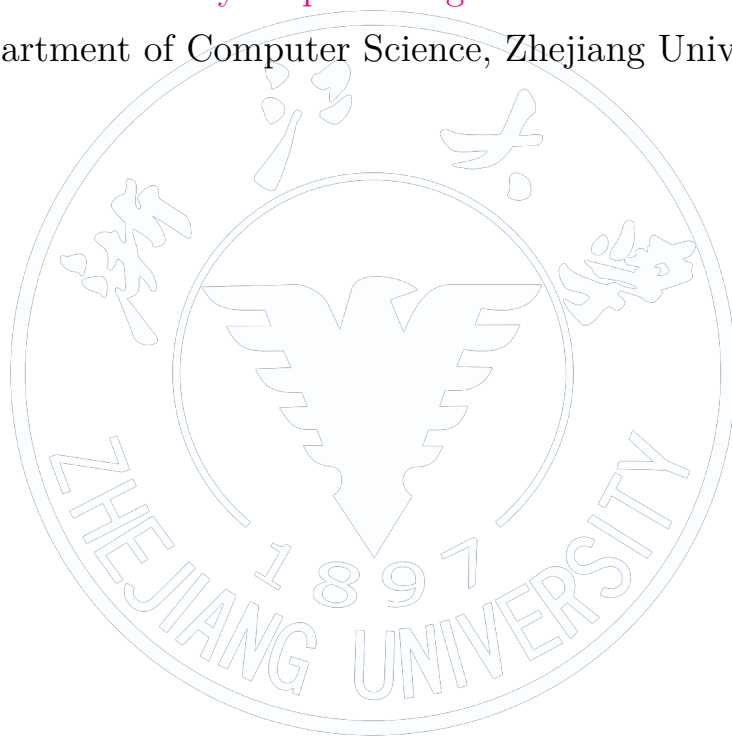
計算機組織

Computer Architecture

TZU-CHUN HSU¹

¹vm3y3rmp40719@gmail.com

¹Department of Computer Science, Zhejiang University



2020 年 12 月 19 日
Version 3.0

Disclaimer

本文「計算機組織」為台灣研究所考試入學的「計算機組織」考科使用，內容主要參考張凡先生的二本計算機組織參考書 [1][2]，以及 wjungle 網友在 PTT 論壇上提供的資料結構筆記 [3]。

本文作者為 TZU-CHUN HSU，本文及其 L^AT_EX 相關程式碼採用 MIT 協議，更多內容請訪問作者之 GITHUB 分頁 [Oscarshu0719](#)。

MIT License

Copyright (c) 2020 TZU-CHUN HSU

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

1 Overview

1. 本文頁碼標記依照實體書 [1][2] 的頁碼。
2. TKB 筆記 [3] 章節頁碼：

Chapter	Page No.
1	1
2	27
3	81
4	101
5	119
6	165
7	221
8	238

3. 省略第一章重點十四，第二章重點五、六、十，第六章重點十四，第七章只看重點一到四及十，第八章重點五一致性協定範例。

2 Summary

1. Theorem (10) Endianness:

- Big Endian: 最左邊或 MSB 放在最低 address, e.g. MIPS。
- Little Endian: 最右邊或 LSB 放在最低 address, e.g. x86。

2. Theorem (47)

- `srl/sll rd, rt, shamt # rs = 5'0`
- `lw/sw rt, imm(rs)`
- `beq/bne rs, rt, addr`
- `addi rt, rs, imm`

3. Theorem (62)

```
int fact (int n) {  
    if (n < 1)  
        return 1;  
    else  
        return (n * fact(n - 1));  
}
```

```
fact:  
    addi $sp, $sp, -8  
    sw $ra, 4($sp)  
    sw $a0, 0($sp)  
    slti $t0, $a0, 1  
    beq $t0, $zero, L1  
    addi $v0, $zero, 1  
    addi $sp, $sp, 8  
    jr $ra
```

```
L1:  
    addi $a0, $a0, -1  
    jal fact  
    lw $a0, 0($sp)  
    lw $ra, 4($sp)
```

```

addi $sp, $sp, 8
mul $v0, $a0, $v0
jr $ra

```

4. Theorem (190) 浮點數:

Single precision		Double precision		Representation
Exponent	Fraction	Exponent	Fraction	
0	0	0	0	± 0
0	$\neq 0$	0	$\neq 0$	\pm denormalized number
1 ~ 254	\times	1 ~ 2046	\times	\pm floating-point number
255	0	2047	0	$\pm \infty$
255	$\neq 0$	2047	$\neq 0$	NaN

5. Theorem (371) 只有 **jump** 和 **MemoReg** 上面 1 下面 0，其他皆相反。

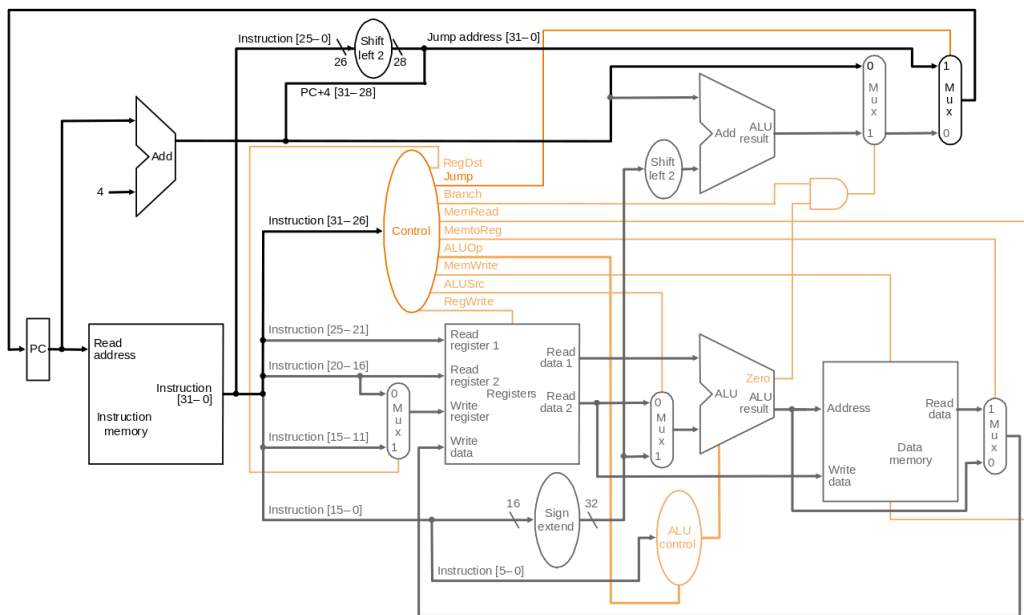


Figure 1: Single-cycle CPU with jump and branch.

6. Theorem (441) 原始 pipeline 設計:

- **beq** 在 *MEM* 決定是否要跳。
- **RegDst** 在 *EX*。

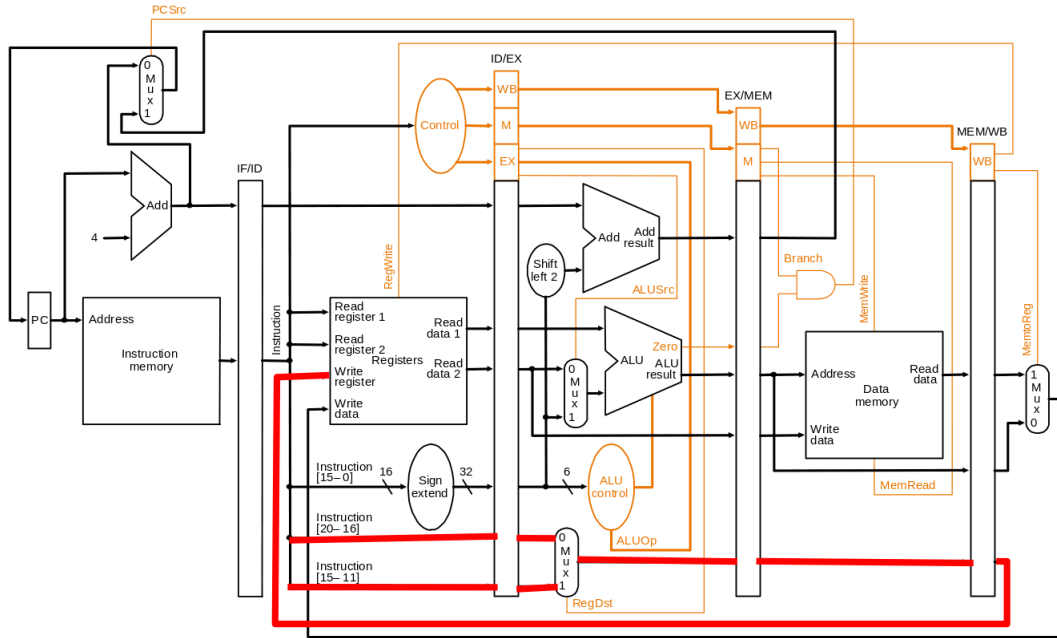


Figure 2: Original pipeline.

7. Theorem (450, 455, 457, 458) Data hazards:

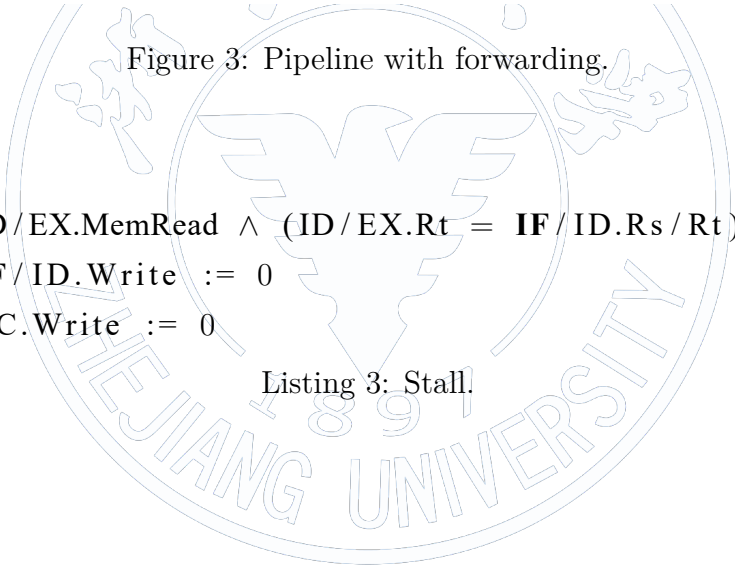
- Forwarding: Combinational units, 放在 EX 因為 ALU。

$\text{if } (\text{EX/MEM.RegWrite} \wedge (\text{EX/MEM.Rd} \neq 0) \wedge$
 $(\text{EX/MEM.Rd} = \text{ID/EX.Rs/Rt}))$
 $\text{ForwardA/B} = 10$

Listing 1: EX hazard.

$\text{if } (\text{MEM/WB.RegWrite} \wedge (\text{MEM/WB.Rd} \neq 0) \wedge$
 $(\neg \text{EX_hazard}) \wedge (\text{MEM/WB.Rd} = \text{ID/EX.Rs/Rt}))$
 $\text{ForwardA/B} = 01$

Listing 2: MEM hazard.



Listing 3: Stall.

- ```

if (ID/EX.MemRead \wedge (ID/EX.Rt = IF/ID.Rs/Rt))
 IF/ID.Write := 0
 PC.Write := 0

```

Listing 3: Stall.

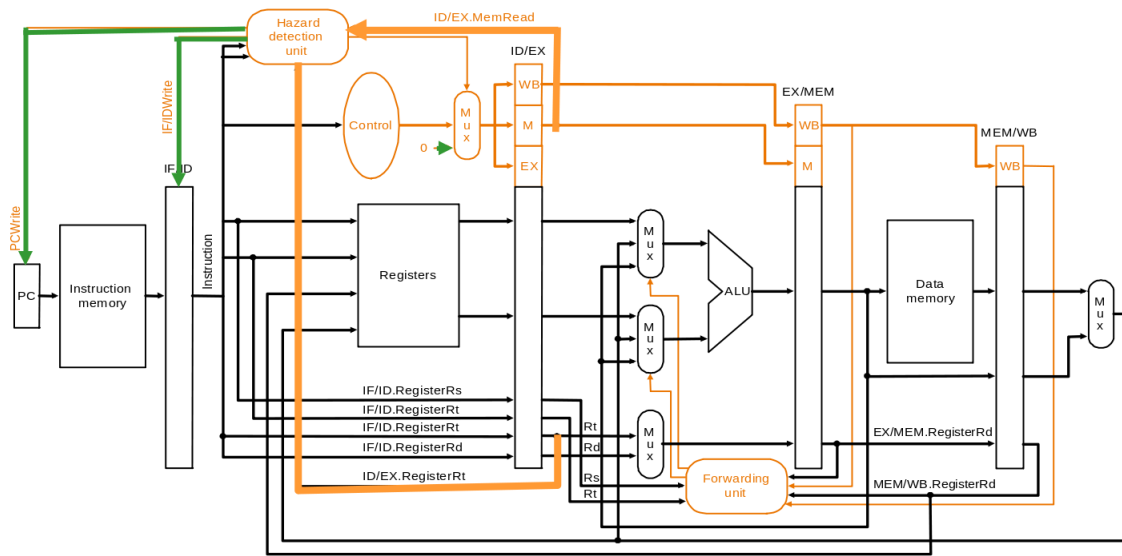


Figure 4: Pipeline with hazard detection and forwarding units.

#### 8. Theorem (478, 487, 494, 559) Control hazards:

- 若分支指令與前一個 ALU 指令或前面第二個 lw 有 data dependency，必須 stall 1 CC。
- 分支指令通過 xor 再 nor 比較是否相同。

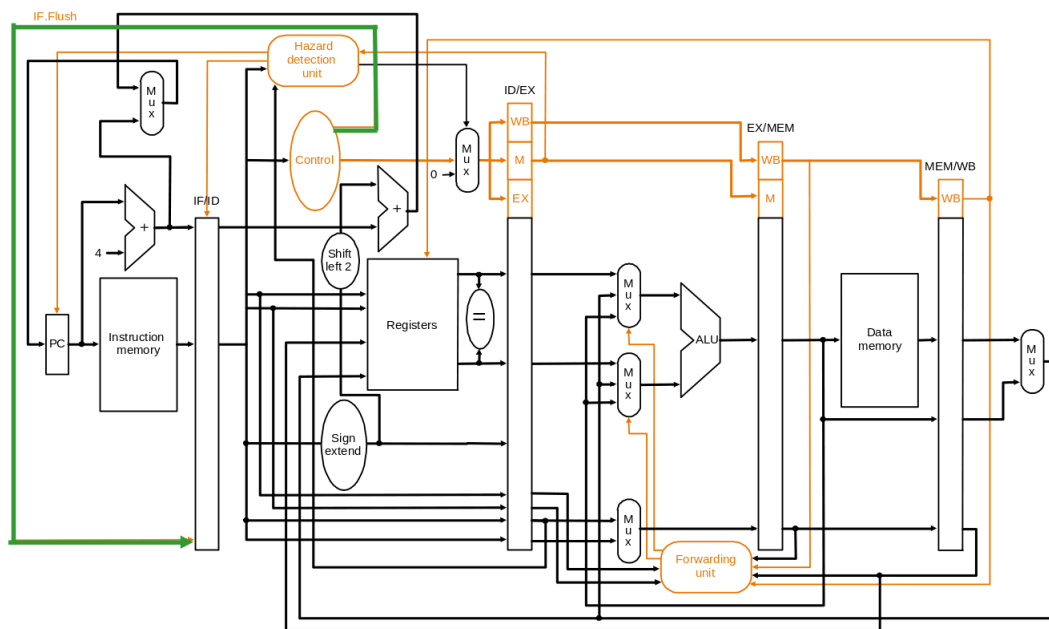


Figure 5: Pipeline with hazard detection, forwarding units and flush.



- Delayed branch:
  - NOT suitable for deep pipeline.
  - From before: 最佳方法，不管跳或不跳皆提升。
  - From target: 用於 branch 發生機率高，有跳才提升。
  - From fall through: 用於 branch 發生機率低，不跳才提升。

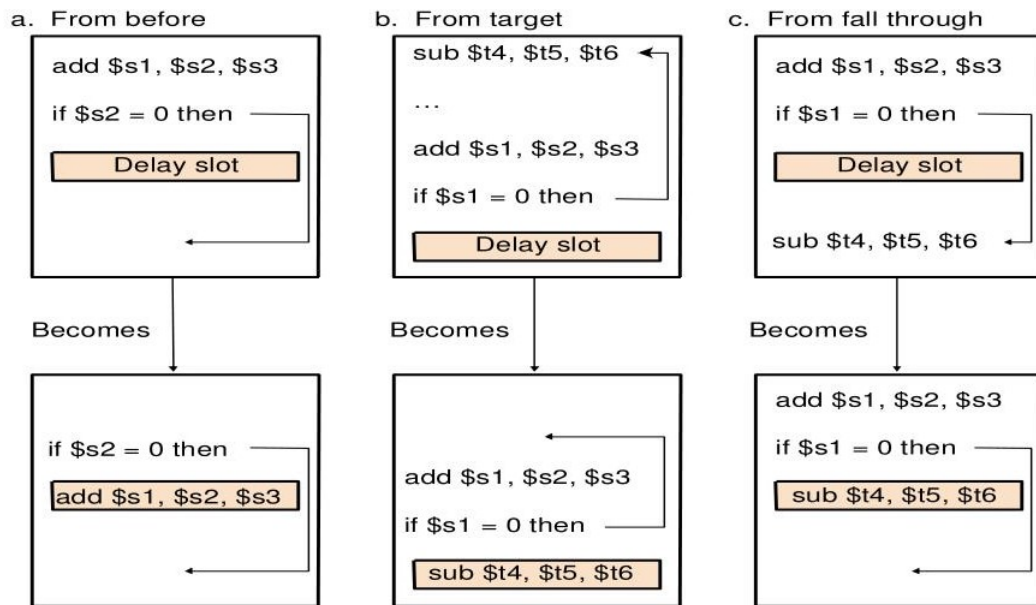


Figure 6: Example of delayed branch.

## 9. Theorem (499, 505, 511)

- Intel IA-64 (EPIC):
  - 支援利用 compiler 開發的平行度。
  - 可以猜測，並利用 if-else 取代 branch。
- Speculation 錯誤復原:
  - 軟體提供修補程式。
  - 硬體 CPU 將猜測結果暫時儲存，若正確，則將猜測結果寫回 register 或 memory，否則 flush buffer。

| Advanced pipeline  |          |          |
|--------------------|----------|----------|
| Technique          | Hardware | Software |
| Branch prediction  | ✓        | ✓        |
| Speculation        | ✓        | ✓        |
| Intel IA-64 (EPIC) | ✓        | ✓        |
| Register renaming  | ✓        | ✓        |
| Prediction         |          | ✓        |

10. **Theorem (27, 48) Cache:**

- Split cache 通常有較差 hit ratio，提升 bandwidth，但不提升 speed。
- L1 cache: 注重減少 hit time; L2 cache: 注重減少 miss ratio。

11. **Theorem (213, 278, 289) RAID:**

- RAID 2: Hamming code, Write 需要讀取所有 disks，從新計算 Hamming code 並寫入 ECC disks，效率差， $2n - 1$  disks。
- RAID 3:
  - Reliability 和 RAID 2 相同。
  - 不做備份，花費較多時間恢復 data， $n + 1$  disks。
  - 當 1 個 disk 出錯可救回來，多個則否。
  - Availability cost 為  $\frac{1}{N}$ ，其中  $N$  為 protection group disks 數量。
  - Parity 集中存放一個 disk。
- RAID 4:
  - 只對 protection group 其中一 disk 做 small reads。
  - $n + 1$  disks，parity 集中存放一個 disk。
  - 當 1 個 disk 出錯可救回來，多個則否。
- RAID 5:
  - Write 就不會有單一 disk 瓶頸。
  - $n + 1$  disks，parity 被分散到所有 disks。
- Read latency: RAID 3 最短; Write latency: RAID 0 或 RAID 3 最短。
- RAID 3 has worst throughput for small writes.
- RAID 3, 4, 5 have same throughput for large writes.

- RAID 4, 5 perform same for parallel small reads and small writes, but RAID 3 can NOT for both access.

## 12. Theorem (320)

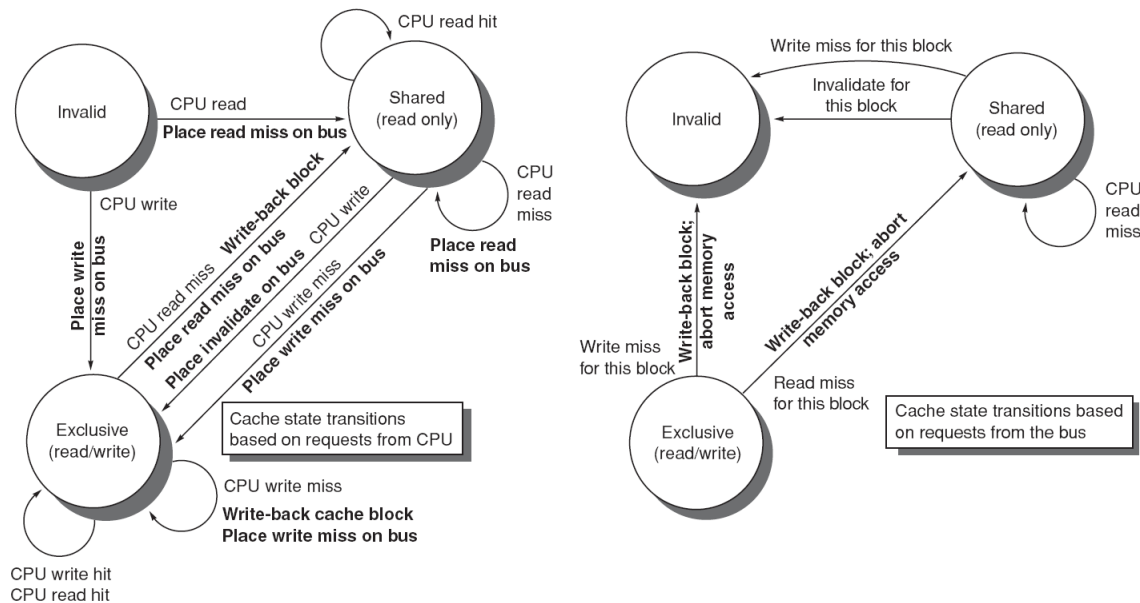


Figure 7: Snooping states.

## 13. Theorem ()

- NUMA is intrinsic in Von Neumann's computer model.
- Physical caches do NOT flush at **context switching**.
- Hyper-threading is **superscalar** and it can speedup **context switching**.
- Out-of-order execution in **cache** level do NOT fail.
- Data fault: Access invalid data memory, which is signaled by **MMU**.
- GPGPU usually runs **SPMT** (Single Program Multiple Thread), and GPU runs **SIMT**.
- L1 data cache is usually separated from L1 instruction cache to **increase bandwidth**.
- Data cache is usually deployed at **MEM** stage.
- Increasing number of **used sticky bits** do NOT improve accuracy.
- **Memory hazard** do NOT cause stall, e.g. **sw** after **lw**.
- Branch target buffer is used by **CPU**.

## References

- [1] 張凡. 計算機組織與結構重點直擊（上）. 鼎茂圖書出版股份有限公司, 3 edition, 2019.
- [2] 張凡. 計算機組織與結構重點直擊（下）. 鼎茂圖書出版股份有限公司, 3 edition, 2019.
- [3] wjungle@ptt. 計算機組織 @tkb 筆記. <https://drive.google.com/file/d/0B8-2o6L73Q2VUkpEMWVLb1pRZE0/view?usp=sharing>, 2017.

