**CSA midterm**

**CP38**

**PartA**

* 1. Computer Architecture คือ Abstractions สำหรับอธิบายการเชื่อมต่อระหว่าง software – hardware
  2. สิ่งที่มีผลต่อการออกแบบ CPU ไม่ทำ เลิก  
     Technology Application  
     Operating System Compiler History

1. Amdahl’s law คือ กฎสำหรับการอธิบาย speedup ของการปรับปรุงการทำงานแบบ sequential  
   สูตร Speedup = 1/(1-p+p/n)  
   มีความเกี่ยวข้องกับ Law of Diminishing returns โดย เมื่อเพิ่มสัดส่วนของส่วนที่ทำความเร็วได้มากขึ้นไปถึงจุดหนึ่งแล้ว Speedup จะลู่เข้าสู่ค่าคงที่ค่าหนึ่ง และไม่สามารถเพิ่ม Speedup อย่างมีนัยสำคัญได้อีกต่อไปแล้ว
2. SPECmark = (v1/t1 \* v2/t2 \* v3/t3 \* v4/t4)0.25  
   Graphical user interface, text

   Description automatically generated
3. fixed cost 400k, Wafer 4k ผลิต die ได้ 1k ชิ้น yield 80%, ค่า test, package อย่างละ 10, test yield 90% ต้องใช้ Wafer อย่างน้อยกี่ชิ้นจึงจะได้ต้นทุนชิ้นละไม่เกิน 75  
   แม่งถามไรวะ ไม่ทำใครจะทำ
4. load 40% ของคำสั่งทั้งหมด ให้ load ใช้ 8 cycles ที่เหลือใช้ 2 cycles  
   ถ้าเปลี่ยน load=4 cycles จะต้องใช้ 1.4Tc ถามว่าควรปรับมั้ย  
   CPI1 = (0.4)(8) + (0.6)(2) = 4.4  
   CPI2 = (0.4)(4) + (0.6)(2) = 2.8  
   Speedup = CPUTime2 / CPUTime1 = 4.4/(2.8\*1.4) = 1.122  
   ควรเปลี่ยนไปใช้แบบที่ปรับแล้ว

**CP39**

**PartA**

**Text

Description automatically generated**

|  |  |  |  |
| --- | --- | --- | --- |
| วิศวกรคนที่ | พิจารณาเฉพาะ A | เฉพาะ C | ทั้ง 3 โปรแกรม |
| 1 | 0.6/(0.6-0.6(0.6)+ 0.6(0.6)/2)  = 1.42857142857 | 0.1/(0.1-0.1+0.1/1)  = 1 | 1/(1-0.6(0.6)+0.6(0.6)/2)  = 1.21951219512 |
| 2 | 0.6/(0.6-0.6(0.6)+ 0.6(0.6)/1)  = 1 | 0.1/(0.1-0.8(0.1)+0.8(0.1)/2)  = 1.66666666667 | 1/(1-0.95(0.3)-0.8(0.1)+0.95(0.3)/2+0.8(0.1)/2)  = 1.22324159021 |
| 3 | 0.6/(0.6-0.3(0.6)+0.3(0.6)/2)  = 1.17647058824 | 0.1/(0.1-0.3(0.1)+0.3(0.1)/2)  = 1.17647058824 | 1/(1-0.3(1)+0.3(1)/2)  = 1.17647058824 |



fa = 1 GHz, CPIa = 2 fb = 0.5 GHz, CPIb = 1  
CPUTimeA = CPITimeB ดังนั้น คอมพิวเตอร์ทั้งสองเครื่องมีประสิทธิภาพเท่ากัน

**CP40**

Table

Description automatically generated

SPEC\_A = 1 SPEC\_B = sqrt(100/10\*1000/100) = 10 SPEC\_C = sqrt(100/40\*1000/20) = sqrt125 > 11

ดังนั้น ประสิทธิภาพที่ดีที่สุดคือ Computer C

Graphical user interface, text

Description automatically generated

3. Good Abstractions = ISA Matrix

- Generalization: ทำงานได้หลายอย่าง

- Orthogonality: ทุกๆ Instruction ใช้หลักการทำงานคล้ายกัน (เช่น Rs, Rt, Rd) (ใช้ Mem หรือ Reg อย่างใดอย่างหนึ่ง)

- Compatibility: backward - Regularity

- Ease of Compilation - Ease of Implementation

4. สิ่งที่ควรบอก compiler developer

- Instructions or Operators

- Instruction format

- Datapath

- Memory Reference (Addressing mode, Memory Organization)

- Interrupt and Exception Handler

5. จงบอกข้อดีข้อเสียของการใช้ register เป็น operand

|  |  |
| --- | --- |
| ข้อดี | ข้อเสีย |
| * เข้าถึงได้อย่างรวดเร็ว * ประหยัดพลังงาน -> ประหยัด cost * ไม่ซับซ้อน | * จำนวนข้อมูลมีได้อย่างจำกัด (โดยทั่วไป 1 Processor มี 32 Register) * การดำเนินการที่ขึ้นกับ Memory จะต้องทำคำสั่ง Load, Store แยกทุกครั้ง หากต้องเรียกใช้ของใน Memory บ่อยจะเสียเวลามาก |

6. memory misalignment ทำให้ access memory มากกว่า 1 ครั้งเนื่องจาก

ข้อมูลอาจจะอยู่ในตำแหน่งระหว่าง 2 ช่องที่ปกติ Bus จะดึงข้อมูลมาได้ เช่น Bus 4 bytes เข้าถึงข้อมูล Address ที่ [n+3:n] ที่ n%4=0 เช่น [7:4] และ [3:0] หากเราวางข้อมูลไว้ไม่ตรงกับตำแหน่งดังกล่าว เช่น วางไว้ที่ [6:3] จะทำให้ต้องดึงข้อมูล 2 ครั้ง

A picture containing graphical user interface

Description automatically generated

ไม่เขียนละกันว่าแปลว่าอะไร Speedup = 1/(1-0.5+0.5/2) = 1.33

**PartB**

1.  
Diagram

Description automatically generated

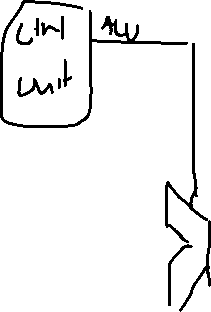
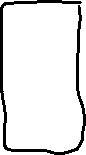
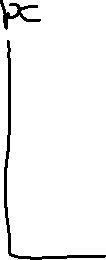
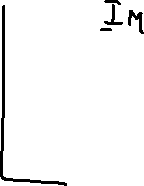
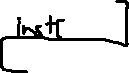
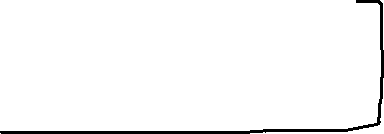
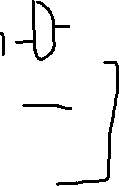
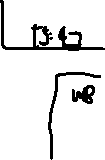
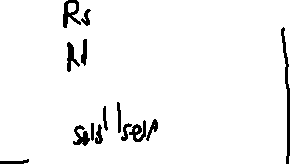
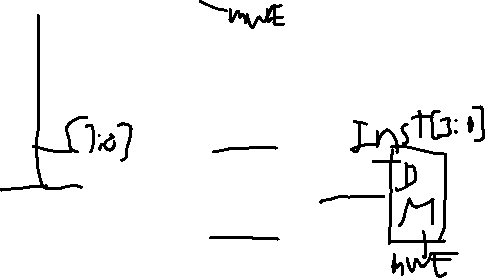
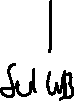
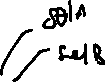
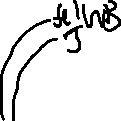
2. Physicals Register Transfer อธิบายถึงการเคลื่อนย้ายของ Data โดยละเอียดเลย เช่น นำค่าอะไรไปใส่ใน Register อะไร ส่วน Logical Data Transfer อธิบายถึงผลลัพธ์จากการทำ Physicals Register Transfer แล้ว เช่น การ Load, Add, Branch

3. Speedup = Single Cycle Processor CPUTime / Multiple Cycle Processor CPUTime

= IC\*CPIa\*Tc / nIC\*CPIb\*Tc  
สังเกตได้ว่า Multiple Cycle Processor จะมีประสิทธิภาพมากขึ้นก็ต่อเมื่อ n \* CPIb > CPIa

Table

Description automatically generated



5.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Sel\_pc | Sel\_addpc | Sel\_b | Sel\_data | Extender\_ops | ALUop | Sel\_wr | Reg\_wr |
| ORI |  |  |  |  |  |  |  |  |
| ORUI |  |  |  |  |  |  |  |  |
| ADD |  |  |  |  |  |  |  |  |
| LW |  |  |  |  |  |  |  |  |
| SW |  |  |  |  |  |  |  |  |
| BEQ |  |  |  |  |  |  |  |  |
| JMP |  |  |  |  |  |  |  |  |

1. เดาว่ามันทำ p RTL

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cycle | ORI rt,rs,imm | ADD rd, rs, rt | LW rt, rs, imm | BEQ rs, rt, imm |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |
| 5 |  |  |  |  |