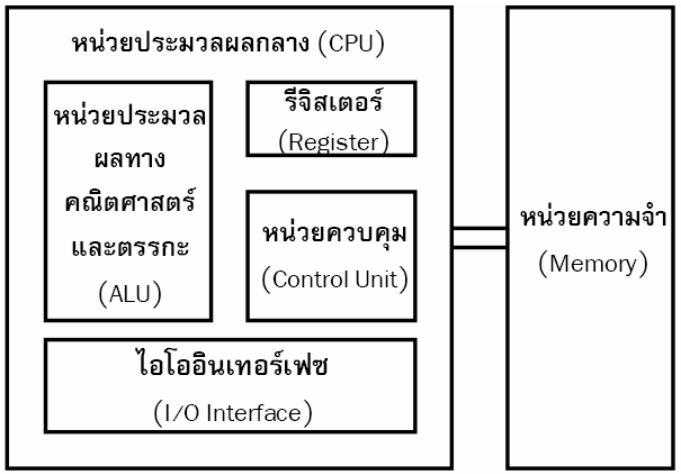
ระบบคอมพิวเตอร์และสถาปัตยกรรม
(Computer System and Architecture)

บทที่ 6
Pipeline, Scalar & Vector Processor

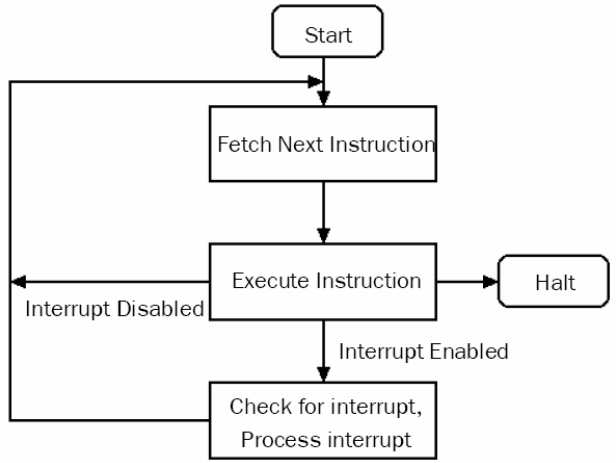
โปรเซสเซอร์

- ปี 1989 Intel ประกาศตัว 80486 ซึ่งเป็นซีพียูแบบ 32 บิต พร้อมเปิดตัวสิ่งที่เรียกว่า "**ไปป์ไลน์**" (Pipeline)
- ไปป์ไลน์ช่วยให้ซีพียูสามารถเฟ็ตช์คำสั่งเข้ามาทำงานได้หลาย ๆ คำสั่งในเวลาเดียวกันได้ โดยเอ็กซิคิวต์ในแต่ละคำสั่งในแต่ละสัญญาณนาฬิกา (Clock cycle) เรียกการทำงานแบบนี้ว่า "**สเกลลาร์**" (Scalar)
- ปี 1993 ได้เปิดตัวซีพียูในยุคที่ 5 ที่เรียกว่า "Pentium" โดยนำไปป์ไลน์มาใส่ไว้ในซีพียูถึง 2 ตัว ทำงานแบบขนานพร้อม ๆ กัน โดยไม่ขึ้นต่อกัน ทำให้สามารถเอ็กซิคิวต์ได้ 2 คำสั่งใน 1 สัญญาณนาฬิกา
- เรียกสถาปัตยกรรมนี้ว่า "ซุปเปอร์สเกลลาร์" (Superscalar)

องค์ประกอบซีพียู



วงรอบการทำงานของซีพียู



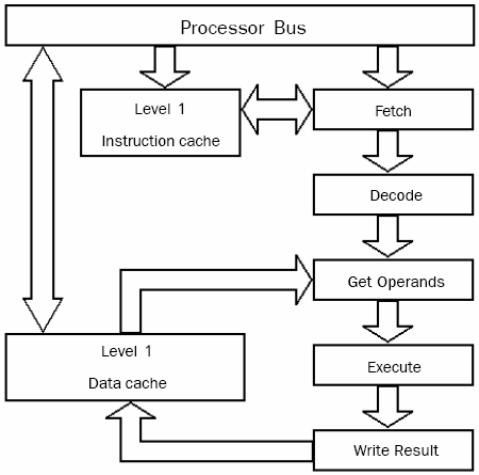
Pipeline

- ไปป์ไลน์ (Pipeline) คือการทำงานแบบคาบเกี่ยวกัน (overlap) โดยการแบ่งซีพียูออกเป็นส่วนย่อย ๆ แล้วแบ่งงานกันรับผิดชอบ
- เดิมไปป์ไลน์เป็นเทคนิคของสถาปัตยกรรมแบบ RISC ต่อมานำมาใช้กับสถาปัตยกรรมแบบ CISC
- แบ่งเป็นภาคหลัก ๆ คือ
  - ภาคเฟ็ตช์คำสั่ง (Instruction Fetch)
  - ภาคการถอดรหัสคำสั่ง (Instruction Decode)
  - ภาครับข้อมูล (Get Operands)
  - ภาคเอ็กซิคิวต์ (Execute)
  - ภาคเขียนผลลัพธ์ (Write Result)

ขั้นตอนการทำงานของ Pipeline

- **ภาคเฟ็ตช์คำสั่ง** หรือ Instruction Fetch ส่วนนี้จะทำหน้าที่รับคำสั่งใหม่ ๆ ทั้งจากหน่วยความจำหลัก หรือจาก Instruction Cache เข้ามา
- **ภาคถอดรหัสคำสั่ง** หรือ Instruction Decode ส่วนนี้จะทำหน้าที่แยกแยะคำสั่งต่าง ๆ ของ CISC
- **ภาครับข้อมูล** หรือ Get Operands ส่วนนี้ทำหน้าที่รับข้อมูลที่จะใช้ในการเอ็กซิคิวต์เข้ามาเก็บไว้
- **ภาคเอ็กซิคิวต์** หรือ Execute ส่วนนี้เป็นขั้นตอนที่ทำการเอ็กวิคิวต์ตามคำสั่งและโอเปอแรนด์ที่ได้รับมา
- **ภาคเขียนผลลัพธ์** หรือ Write Result เมื่อทำการเอ็กซิคิวต์เสร็จเรียบร้อยแล้ว ผลลัพธ์ที่ได้ก็จะนำไปเก็บไว้ในรีจิสเตอร์ หรือในแคช

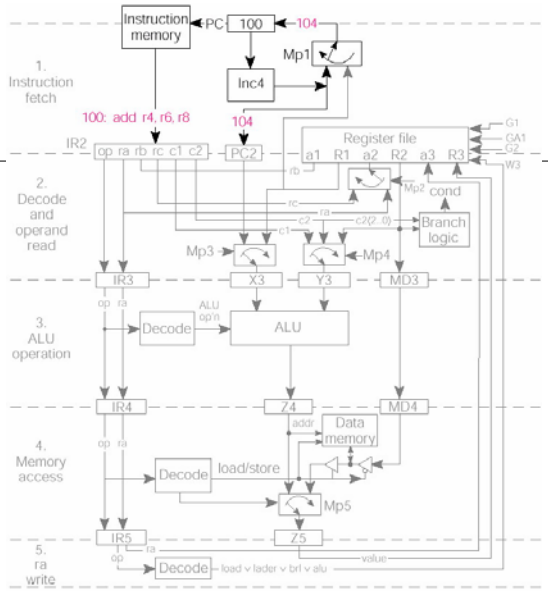
ขั้นตอนการทำงานของ Pipeline

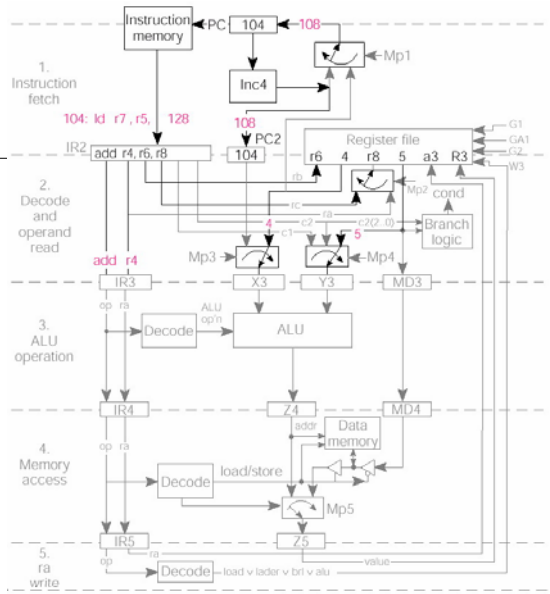


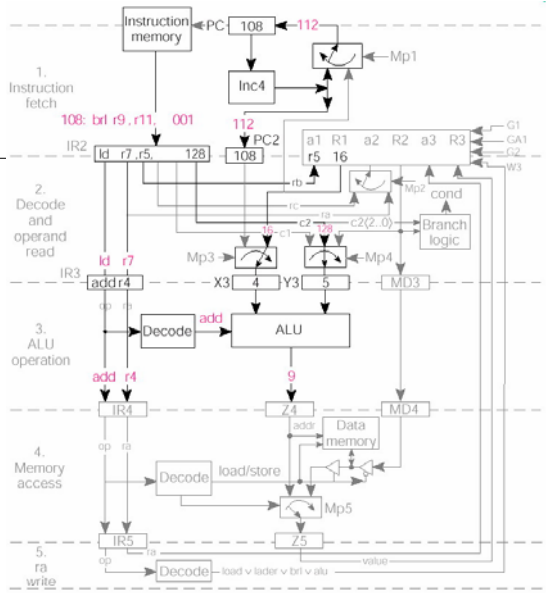
การทำงานของ Pipeline

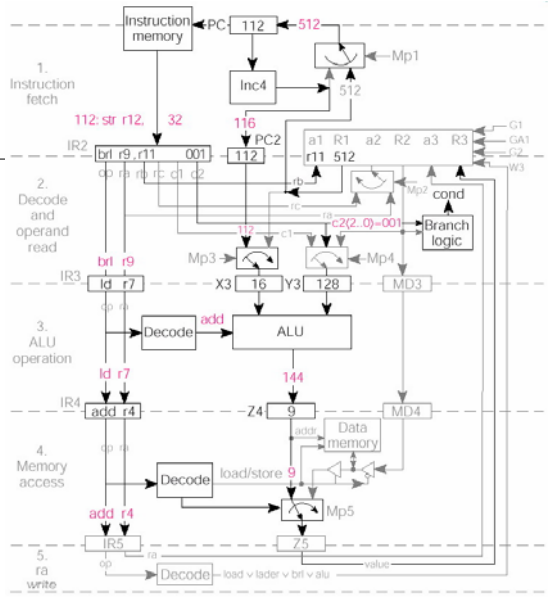
```
100 :   add r4, r6, r8      ; R[4] <-- R[6] + R[8]
104 :   ld r7, 128(r5)      ; R[7] <-- M[R[5] + 128]
108 :   brl r9, r11, 001    ; PC <-- R[11] (=512) : R[9] <-- PC
112 :   str r12, 32         ; M[PC+32] <-- R[12]
        ...
512 :   sub …               ; Next instr …
```

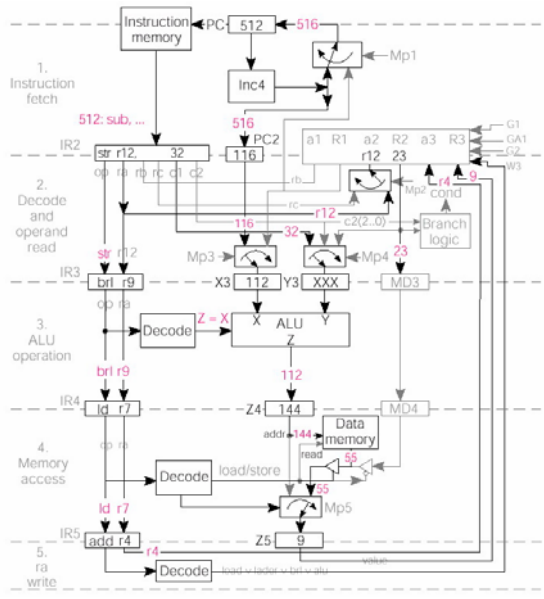
การทำงานของ Pipeline เทียบกับชุดคำสั่ง Assembly











Pipeline hazard

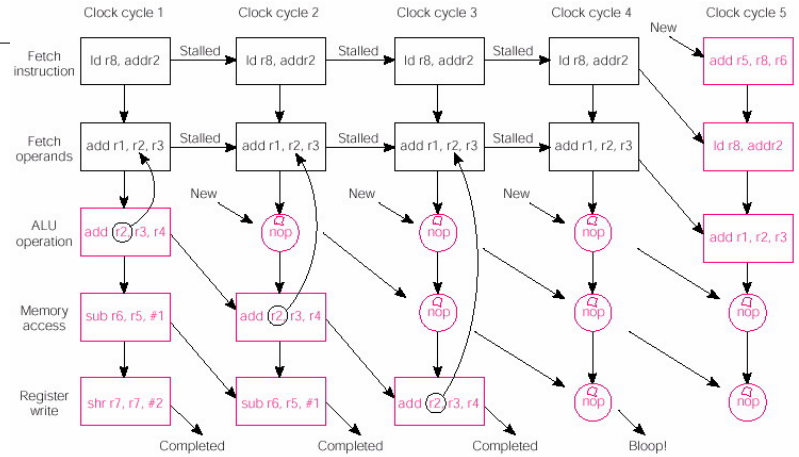
- เหตุการณ์ที่อาจเกิดผลกระทบอย่างรุนแรงกับผลลัพธ์ที่ออกมาได้ เนื่องจากคำสั่งบางคำสั่งจะมีการเขียนผลลัพธ์ลงบนตัว Operand บางตัวที่ต้องถูกอ่านค่าจากอีกคำสั่งหนึ่ง
- ประเภทของการเกิด Pipeline hazard
  - Data hazard
  - Branch hazard

Data hazard

- “Read after write” หรือ RAW Data hazard

```
100: add r0, r2, r4
104: sub r3, r0, r1
```

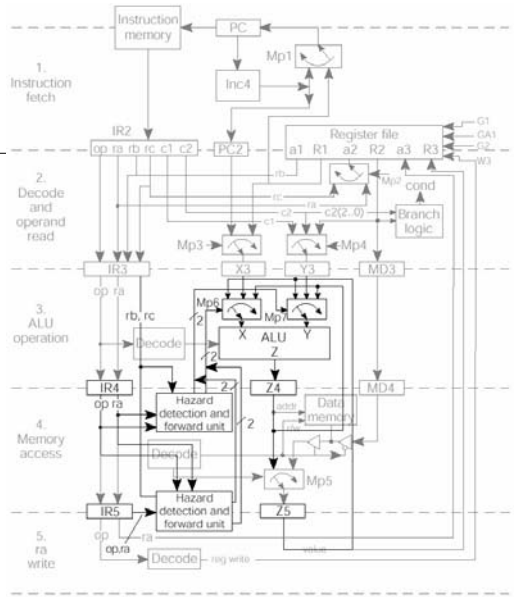
Data hazard



Data hazard

- Write after write (WAW)
- Write after read (WAR)
- เปรียบเทียบ RAW, WAW และ WAR

RAW	WAW	WAR
1. add r0, r1, r2	1. add r0, r1, r2	1. add r2, r1, r0
2. sub r4, r3, r0	2. sub r0, r4, r5	2. sub r0, r3, r4

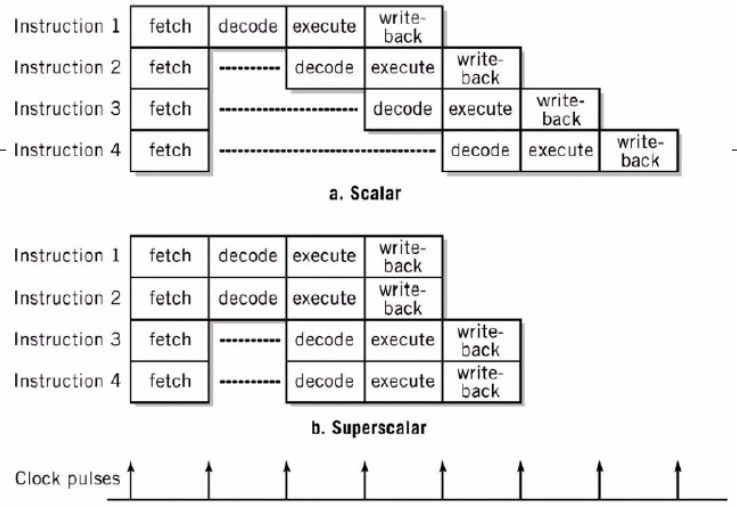


Branch hazard

- Branch delay
- ระบบจะต้องเก็บข้อมูลบางอย่างเอาไว้เช่นแอดเดรสของชุดคำสั่งก่อนและหลังที่จะพบปัญหา branch delay หรือค่า PC ของชุดคำสั่งถัดจากคำสั่ง branch ควรจะเป็นค่าใด
- หน่วยความจำแคช มักจะใช้ branch prediction ที่เรียกว่า Branch Target Buffer (BTB) เข้ามาทำงานร่วมกันในชิพของโปรเซสเซอร์

Scalar & Superscalar Processor

- โปรเซสเซอร์แบบ “สเกลลาร์” (Scalar processor) ใช้การประมาณค่าเฉลี่ยความเร็วของการเอ็กซิคิวต์คำสั่งให้เท่ากับความเร็วของสัญญาณนาฬิกา (clock speed)
- โปรเซสเซอร์แบบ “ซุปเปอร์สเกลลาร์” (Superscalar processor) ใช้การเอ็กซิคิวต์คำสั่งแบบคู่ขนานกันไป ดังนั้นในหนึ่งสัญญาณนาฬิกาจะสามารถเอ็กซิคิวต์คำสั่งได้มากกว่าหนึ่งคำสั่ง
- ระบบไปป์ไลน์และซุปเปอร์สเกลลาร์ ไม่ได้หมายความว่าทำให้ขั้นตอนการเอ็กซิคิวต์คำสั่งลดน้อยลง (การเอ็กซิคิวต์คำสั่งยังคงมี 5 ขั้นตอน) เพียงแต่สามารถที่จะทำงานแบบคู่ขนานกันได้ ส่งผลให้ในเวลาที่เท่ากัน

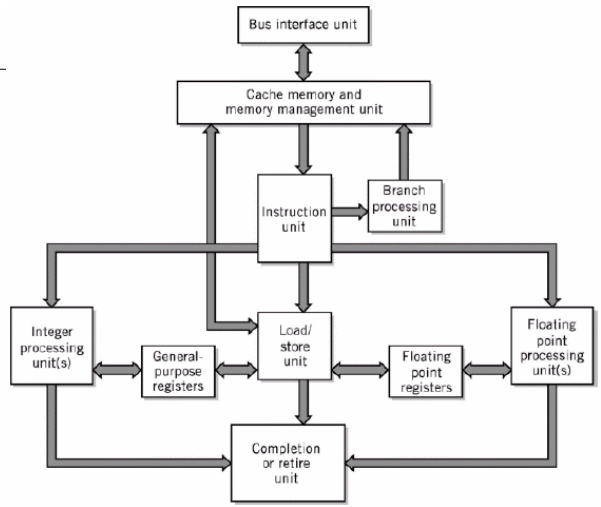


การประมวลผลแบบสเกลลาร์ และซุปเปอร์สเกลลาร์

ปัญหาการเอ็กซิคิวต์ของ Superscalar Processor

- ปัญหาที่เกิดจากลำดับก่อนหลังในการประมวลผลคำสั่งเสร็จสิ้น
- ปัญหาที่เกิดจากการเปลี่ยนทิศทางการทำงานของโปรแกรมเนื่องมาจากคำสั่ง branch
- ปัญหาการขัดแย้งกันในการใช้ทรัพยากรภายในซีพียู เช่น general-purpose register

Modern Superscalar

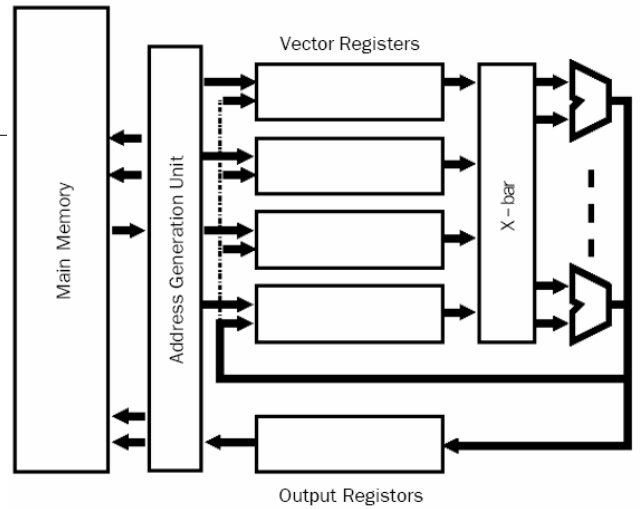


เวกเตอร์โปรเซสเซอร์ (Vector Processor)

- ทั่วไปเรียกว่า “ซุปเปอร์คอมพิวเตอร์” (Supercomputer)
- ใช้สถาปัตยกรรมไปป์ไลน์ที่ประมวลผลแบบเวกเตอร์และเมทริกซ์ทำให้ระบบมีประสิทธิภาพสูง
- โปรเซสเซอร์นี้สามารถประมวลผลเวกเตอร์ (ชุดของข้อมูล) ทั้งชุดได้โดยคำสั่งเดียว
- โอเปอแรนด์จะเป็นเวกเตอร์ เช่น C=A+B เป็นผลบวกของข้อมูลในเวกเตอร์เป็นคู่ ๆ
- มีรีจิสเตอร์ในตัวโปรเซสเซอร์ที่เรียกว่า “เวกเตอร์รีจิสเตอร์” (Vector Register)

เวกเตอร์โปรเซสเซอร์ (Vector Processor)

- ข้อมูลที่อ่านเข้ามาในเวกเตอร์โปรเซสเซอร์จะเป็นคิวแบบ FIFO (First-In First-Out)
- ชุดคำสั่งประกอบด้วยคำสั่งที่สามารถ
  - โหลดเวกเตอร์รีจิสเตอร์จากตำแหน่งในหน่วยความจำ
  - ทำโอเปอเรชันต่าง ๆ ในเวกเตอร์รีจิสเตอร์
  - เก็บค่าข้อมูลจากเวกเตอร์รีจิสเตอร์กลับลงหน่วยความจำ
- การคอมไพล์จะมี Vectorizing Compiler เป็นตัวคอมไพเลอร์
- แปลงการวนรอบ ให้กลายมาเป็นคำสั่งแบบเวกเตอร์เพียงคำสั่งเดียว



โครงสร้าง Vector Register

เวกเตอร์โปรเซสเซอร์ (Vector Processor)

- ประเภทของ Vector Processor
  - Memory-to-memory Vector Processor
  - Register-to-register Vector Processor
- Memory-to-memory Vector Processor มีความสามารถในการประมวลผลเวกเตอร์ที่ยาวมาก ๆ มีข้อเสียคือมีค่าเสียเวลาที่เรียกว่า startup time สูง ประสิทธิภาพของเวกเตอร์โปรเซสเซอร์ จะอยู่ในรูป

  $$T = s + aN$$

- Register-to-register Vector Processor แบ่งเวกเตอร์ออกเป็นชิ้นย่อย ทำการคำนวณเป็นชุด ๆ

เวกเตอร์โปรเซสเซอร์ (Vector Processor)

- Cary Supercomputer
- Vector Chaining ของ Cary Supercomputer

  V2 = V0 * V1

  V4 = V2 + V3

Vector Chaining