1 Vector Processing Requirement

ในบทนี้เราจะอธิบาย Concept ง่ายๆของ vector processing และ เครื่องมือที่จำเป็น เราสามารถแบ่ง vector processing โดยรูปแบบของ Sealar Processing อธิบายลักษณะเฉพาะของคำสั่ง Vector และกำหนดประสิทธิภาพในการวัดของ vector processor เราศึกษา Vector เพื่อเป็นมาตราฐานการประมวล , เป็นลักษณะพิเศษของคำสั่ง Vector และกำหนดประสิทธิภาพการวัดของคำสั่ง Vector และ Vector เป็นวิธีในการแนะนำ pipeline computer

1.1 Characteristics of Vector Processing

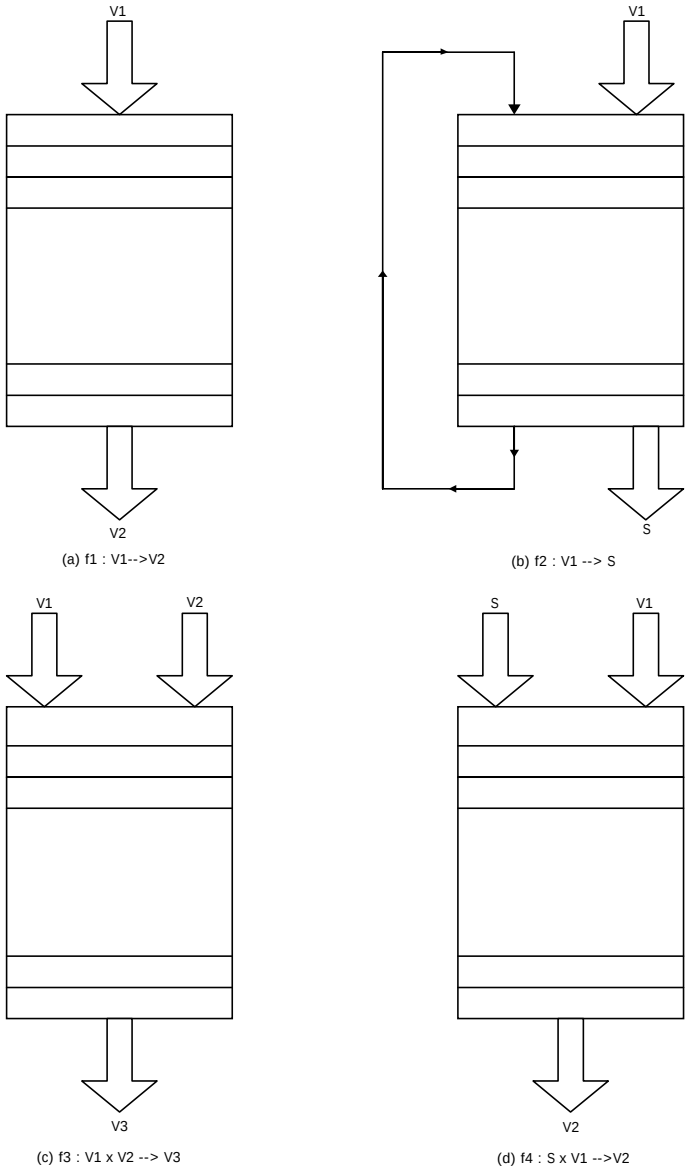
Operand ของ Vector เป็น set ของ n element โดย n คือ ความยาวของ Vector ตัวแปลใน Vector อาจจะเป็น ทศนิยม, จำนวนเต็ม , Boolean , อักขระ ซึ่ง Vector สามารถกำหนดทั้ง 4 ชนิดนี้เป็นตัวแปลได้ ลักษณะพิเศษเหล่านี้ของข้อมูล Vector มันใช้งานได้ง่าย เช่น Boolean สามารถที่จะเปรียบเทียบค่าของ 2 Vector สามารถใช้หรือไม่ใช้ Masking Vector เป็นองค์ประกอบของ Operation ในคำสั่ง Vector

คำสั่ง Compress เป็น Vector สั่นๆ ภายใต้การควบคุมของ Masking Vector

คำสั่ง Merge เป็นการรวมกันของ 2 Vector ภายใต้การควบคุมของ Masking Vector

Table 1 some representative vector instructions

Type	Mnemonic	Description (l = 1 through N)	
F1	**VSQR**	**Vector square root:**	**B (l) ← A (l)**
	VSIN	**Vector sine:**	**B (l) ← sin (A (l))**
	VCOM	**Vector complement:**	**A (l) ← A (l)**
F2	**VSUM**	**Vector summation:**	**$S = \sum^{N}_{l=1} A(l)$**
	VMAX	**Vector maximum:**	**$S = Max_{l=1,N} A(l)$**
F3	**VADD**	**Vector add:**	**C (l) = A (l) + B (l)**
	VMPY	**Vector multiply:**	**C (l) = A (l) * B (l)**
	VAND	**Vector and:**	**C (l) = A (l) and B (l)**
	VLAR	**Vector larger:**	**C (l) = Max (A (l), B (l))**
	VTGE	**Vector test >:**	**C (l) = 0 if A (l) < B (l)**
			C (l) = 1 if A (l) > B (l)
F4	**SADD**	**Vector-scalar add:**	**B (l) = S + A (l)**
	SDIV	**Vector-scalar divide**	**B (l) = A (l)/S**



Compress และ Merge เป็นเครื่องหมายพิเศษ (f1,f2) เพราะตัวแปรของ output มีขนาดแตกต่างกับตัวแปร input ในเครื่อง computer ทั่วไป จะมีตารางความจริงสำหรับ pipeline จะมีลักษณะดังนี้

A. Identical processor คือ การขอเวลาทำซ้ำ

B. Successive Operand

C. เมื่อ Executed operation โดย pipeline จะมีการ share ทรัพยากรรวมกัน

ลักษณะพิเศษอธิบายว่าทำไม Vector Processor ถึงมีโครงสร้าง Pipeline คำสั่ง Vector ใช้ Field ต่างๆต่อไปนี้

1. Operation Code เป็นรายละเอียดที่ใช้ในการเลือก function เดียว หรือ function รวมในการทำ operation
2. Vector register จะเป็นคำสั่งสำหรับอ้างลง memory
3. Address increment
4. Address effect เกี่ยวข้องกับ base address และ effect ซึ่งตำแหน่งของ memory สามารถคำนวณได้ offset เป็นได้ทั้ง + และ –
5. Vector length คือ การกำหนดจุดสิ้นสุดของคำสั่ง Vector

เราสามารถแบ่ง Pipeline ของ Vector ออกเป็น 2 สถาปัตยกรรม

1. memory-to-memory ใน source operands ตรงกลาง และสุดท้ายของผลลัพธ์ถึงมาจาก main memory สำหรับ memory-to-memory ทั้ง 5 ข้อที่กล่าวมาข้างต้น จะบอกรายละเอียดของคำสั่ง data ระหว่าง main memory กับ pipeline
2. register-to-register ใน operand และผลลัพธ์ถึงมาจาก main memory ส่งผ่านไปให้ scalar register

Vector-length register สามารถใช้ควบคุม Vector operation ใน Vector processing ใช้ Pipeline Overhead ของ pipeline processing เป็นส่วนสำคัญของ setup time ซึ่งเป็นเส้นทางระหว่าง operands กับ functional units overhead อื่นๆ เช่น Flushing time เป็น decoding ของคำสั่ง Vector และเป็นการออกของผลลัพธ์แรกของ pipeline Flushing time เกิดจาก Vector และ Scalar processing อย่างไรก็ตาม Vector pipeline มีการ Check เงื่อนไขสุดท้ายและควบคุม Vector

ความยาวของ Vector มีผลกระทบต่อการ process เพราะว่า Vector ยาวอาจจะทำให้ overhead สูง

<u>Enrich the vector instruction set</u>

Instruction set ที่สูงขึ้น ทำให้การ process สูงขึ้น และทำให้มีการ access memory เพิ่มขึ้น การที่ instruction มากขึ้นทำให้ใช้ประโยชน์จากทรัพยากรได้ไม่สมบูรณ์

<u>Combine scalar instruction</u>

ใช้ Pipeline สำหรับ processing scalar จำนวนมาก

<u>Choose suitable algorithms</u>

หลายๆครั้งที่มี Algorithm ที่ทำงาน เร็วนั้น สนับสนุนการเรียงลำดับ processor แต่อาจจะไม่เป็นแบบนั้นใน pipeline processor

Use a vectorizing compiler

Compiler ที่ฉลาดๆ จะพัฒนาถึงการตรวจพบ Vector instruction การที่ Vectorizing compiler ใช้ภาษา sequential ทำให้ parallelism เกิดน้อย

ภาษา Sequential คือการใช้ High-level programming languages ในการสร้าง parallel บน Vector processor

จาก 4 ข้อบนนี้ กลายเป็นการพัฒนาของ Parallelism ใน advanced programming

Parallel algorithm (A)

High-level languages (L)

Efficient object code (O)

Target machine code (M)

Degree ของ parallelism อ้างถึงตัวเลขที่เป็นอิสระต่อกัน สามารถปฏิบัติพร้อมกันได้ เราต้องการค้นหา suitable algorithm ของ high parallelism เพื่อแก้ปัญหา large-scale matrix เราต้องพัฒนา parallel language เพื่อให้เป็น high-level language ไม่มีมาตราฐานของ parallel language ที่เป็นที่ยอมรับ ปัจจุบัน User ส่วนใหญ่ ใช้ sequential language

1.2 Multiple Vector Task Dispatching

รูปแบบชิ้นงานของ Parallel เป็นการเสนอ multi-pipeline Vector processor รูปแบบสามารถประยุกต์ หาค่ามากที่สุดใน Vector supercomputer รูป 3.43 เป็นโครงสร้างสำหรับของ Vector processor main memory คือ ส่วนที่เข้าบ่อยจึงใช้เวลาในการ access น้อย คำสั่งและข้อมูลอาจจะปรากฏใน Vector หรือ scalar Formats Instruction processing unit (IPU) fetch และ decode ทั้ง scalar และคำสั่ง Vector ทุกคำสั่งของ scalar จะใช้ scalar processor ในการ execution ในตัว Vector processor จะมี multiple scalar pipeline

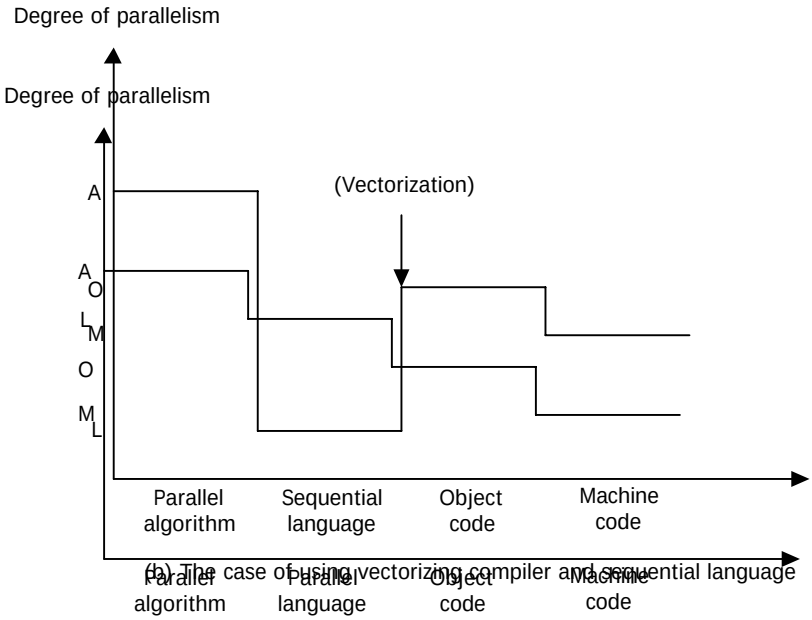


Figure 3.42 Parallelism regeneration in using a vectorizing compiler for programs writen in sequential language.
(a) The idea case of using parallel algorighm/language

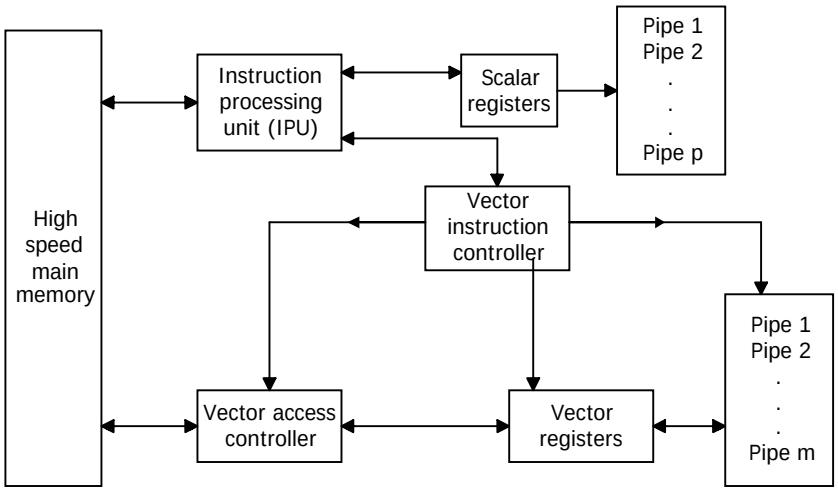


Figure 3.43 The architecture of a typical vector processor with multiple functional pipes.

ภายหลังจากคำสั่ง Vector ทำงานผ่าน IPU Vector instruction controller จะรับช่วงต่อ มาเพื่อทำการ execution function จากทำงานจะถอดรหัส คำสั่งของ Vector คำนวณ address ของ effective Vector-operand ,setting up Vector access controller และ Vector processor และจะจับตรวจสอบการประมวลผลของคำสั่ง Vector รูปแบบของ Vector เริ่มจากการอธิบายโครงสร้างพื้นฐาน จากการติดต่อ machine vector access controller จะรับผิดชอบในการ fetch vector operand โดยเรียงจากการเข้าถึง main memory Vector Register ใช้ close up speed gap ระหว่าง main memory กับ Vector processor

1.3 Pipelined Vector processing Methods

เราสามารถแบ่งวิธีในการ Process vector ได้ 3 ชนิด

1. Horizontal processing โดยจะคำนวณประสิทธิภาพจาก ซ้าย->ขวา ในแบบ row ทุกๆองค์ประกอบของ Vector Y เป็นการคำนวณในลำดับ Y_i โดย i .. m ผลรวม $Y_I = \sum^{N}_{j=1} Z_{ij}$

$$T_a \text{ (horizontal)} = (mn + 14m)r$$

เป็นวิธีการที่ใช้บอกความถี่ใช้ scalar pipeline processor speed up ของ horizontal pipeline บน vector processor เรียงการ process ดังนี้

$$S_{horizontal} = 10mn – sm / 2mn + 14m + 9$$

2. Vector processing ใช้ในการคำนวณ carried out จากบนลงล่างตาม calumn
3. Vector looping จำคำนวณทั้ง ซ้าย->ขวา และ บนลงล่าง เป็นการรวมกันทั้ง Horizontal processing กับ Vector Processing

2.1 early vector processor

คอมพิวเตอร์พิเศษสำหรับ vector processor เริ่มทำงานด้วยระบบพื้นฐาน 2 ระบบของ supercomputer ซึ่งเป็นที่รู้จักกันในนาม CDS-Star และ TI-ASC, ซึ่งทั้งคู่มีการจัดการหลายรูปแบบสำหรับ stand-alane operations ซึ่งเป็นรากฐานของการใช้เทคโนโลยีและการออกแบบแผนและการติดต่อกันและกันของไมโครโปรเซสเซอร์, vector processor ทั้ง 2 รุ่น แตกต่างกันหลายอย่างในส่วนนี้จะกล่าวถึง โครงสร้างของการออกแบบแผนและการติดต่อกันและกันของไมโครโปรเซสเซอร์ การออกแบบการส่งข้อมูลด้วยตัวเลข และ vector processor ใน the star และ ASC ในส่วนต่อไปนี้ เราจะเรียนเกี่ยวกับ vector processor ในปัจจุบัน

2.1Architecture of star-100 and IT-ASC ประกอบด้วยระบบการส่งข้อมูลด้วยตัวเอง 2 ส่วนที่ต่างกัน .control data corporation เริ่มต้นออกแบบ star ในปี 1965 และผลิตได้ในปี1973 มันคือโครงสร้างรอบๆ a four million byte eight million optional ซึ่งเป็นหน่วยความจำที่กว้างมากสำหรับ stan-alone operations ลักษณะพิเศษของ star-100 ประกอบด้วย stream processor, virtual addressing,hardware misroinstructions,semiconductor memory-registerfile และ pipelined floating-point arithmetic. หน่วยความจำใน the star มี cycle time 1.28 us มันมีถึง 32 หน่วยความจำซึ่งแต่ละหน่วยบรรจุ 2048 ตัวอักษร 512 bit. The memory cycle ซึ่งถูกแบ่งออกเป็น32 minor cycle , ด้วยอัตรา 40 nanosec ต่อรอบวง ซึ่งหมายความว่าหน่วยความจำส่งข้อมูล512 บิท / 1 วงรอบ

The pipelined arithmetic unit ถูกออกแบบมาโดยเฉพาะเพื่อการทำงานที่ต่อเนื่อง และ มีทิศทางเดียวกัน บน single bits, 8-bit byte, และ 32-บิท หรือ 64-บิท vertual address ใช้เทคนิคhigh-speed mapping เพื่อการเปลี่ยนแปลง จาก logical address เป็น absolute memory address

ในกรณีอุดมคตินี้ ระบบมีประสิทธิภาพในการผลิต 100 million 32-บิท floating-point / วินาที ระบบโครงสร้างของ the star-100 ให้ดูในรูป4.2 หน่วยความจำได้ถูกสร้างให้มี 8 กลุ่ม ของ 4หน่วยความจำ ในระหว่างการทำงาน โดยแต่ละบัสจะทำการย้ายข้อมูลโดยอัตรา 128 บิท ต่อ 1 วงรอบ โดยบัส 2 ตัว ถูกใช้เพื่อ ส่ง operand streams ไปยัง pipeline processor บัสตัวที่3 ถูกใช้เพื่อเก็บรักษา streamทั้งหมด และ บัสตัวที่4 ถูกใช้เพื่อ แบ่งส่วนระหว่าง input-output storage requests กับ references of control vector

The Storage Access Control Unit ควบคุม การส่งผ่านของข้อมูลไปยังหน่วยความจำ มันมีหน้าที่แบ่งหน่วยความจำระหว่าง บัสทั้งหลายด้วย stream และ I/O unit function ที่สำคัญที่สุดของมันคือการจัดการ virtual memory address เกี่ยวกับการเปรียบเทียบและโยกย้าย The stream unit จัดหาการควบคุมพื้นฐานสำหรับระบบหน่วยความจำอ้างอิงทั้งหมด และสัญลักษณ์การควบคุมทั้ง

หลายเริ่มจาก The stream unit มันมีการถอดรหัสด้วยความรวดเร็ว The read buffer และ white buffer ถูกใช้เพื่อให้บัสทั้ง 4จัดส่งข้อมูลให้พร้อมกัน The memory requests … สุดท้ายนี้ อัตราการส่งข้อมูสที่เร็วที่สุดสามารถแบ่ง address ของ 4บัสได้

ฟังก์ชันอื่นๆใน The stream unit ประกอบด้วย แฟ้มการจดทะเบียน และ รหัสหน่วยความจำ แฟ้มการจดทะเบียนบรรจุ addressing ที่สำคัญสำหรับการเริ่มทำงานและทำงานเสร็จ มันมีความสามารถทำงานด้วยระบบ logical และ arithmetic . The semimicrocode memory ถูกใช้ในส่วน stream condition ถูกสร้างโดย microcode พร้อมกันกับ the hardware control เพื่อดำเนินการแนะนำและขัดขวาง the string unit ดำเนินการจัดเรียงด้วยระบบเลขฐาน 10 หรือเลขฐาน2 และดำเนินการ bit-logical และกระทำการเรียงตัวอักษร มันบรรจุด้วยตัวบอกหลายตัว เพื่อดำเนินการด้วย binary code decimal(BCD) และ binary arithmetic

The star-100 คือ arithmetic pipelines ที่อิสระ 2 เส้นทาง ดู(รูป4.5) The pipeline processor ประกอบด้วย 64 bit floating point henceforth FLP เพิ่ม unit และ 32 bit FLP multiply unit .pipeline ที่เพิ่มมาทางด้านขวา ประกอบไปด้วย 4 ส่วน ในส่วนของ The exponent compare เปรียบเทียบการอธิบายและเก็บส่วนที่ใหญ่ไว้ ความแตกต่างระหว่าง The exponent คือ ในส่วนที่เพิ่ม The shifted และ unshifted ได้ถูกเพิ่มเข้ามา จำนวนรวมและ The exponent ที่ใหญ่ที่สุดจะผ่านไปยังส่วนของ normalized ส่วน transmit จะทำการเลือกสิ่งที่ต้องการ upper or lower half of sum ตรวจสอบ function ที่เกินออกมา และส่งผลลัพธ์ไปยัง data bus ที่ระบุไว้ นี้คือ เส้นทางจาก output ของส่วนจัดส่งไปยัง input ของส่วน receive ผลตอบรับนี้ มีประโยชน์สำหรับ continuous addition of multiple floating-point number อย่างไรก็ตามเมื่อ nonstreaning-type operation ได้ถูกดำเนินการ เวลาที่ใช้ในการจัดการจะลดลง 50% ถ้า output จาก operation จำเป็นสำหรับ input operand เพื่อ sub sequent operation

ในส่วนของ hardware ที่เพิ่มเข้าไป มันมีความเป็นไปได้ที่จะแบ่งแยก 64 bits เพิ่มเข้าไปใน pipeline เป็น 2 ส่วน ส่วนละ 32 bit เพราะฉะนั้น ครึ่งหนึ่ง(32 bit) arithmetic สามารถหามาได้ง่าย The 32 bit multiply pipeline ถูกทำให้สำเร็จด้วย multiplier recording logic,multiplicand-getting network และ severol levels of carry-save address ผลลัพธ์ของการผลิต multiplication

processor number 2 รูป 4.6 b ประกอบด้วย pipline add unit , a non pipelined devide unit, a pipeline multi punpose unit และ some pipelined merge unit การเพิ่ม pipeline .o processor number 2 ก็เหมือนกับ processor number . The multipurpose pipeline มี 24 ส่วน และ มีความสามารถในการเพิ่มจำนวนลดจำนวน , root , และจำนวนของ arrrrithmetic logic operetion อื่นๆ The regiater devide unit คือ nonpipelined devider ซึ่งสามารถดำเนินการด้วย BCD arithmetic.

two 32-bit multiply pipeline สามารถประกันเพื่อให้กลายเป็น 64-bit multiply pipeline การประกอบกันนี้สามารถปฏิบัติได้พร้อมกัน 32-multiplicatons หรือ 64-multiplications ในคำสั่งที่จะดำเนินการ 64-bit multiplication เลขจำนวนที่ถูกคูณ A และตัวคูณ B ได้แยกเป็น 2 ส่วน A = A0 +A1 x 2W , และ B = B0 + B1 x 2W เมื่อ W = 32 bits ความกว้างของ baisc multiple pipeline ดังนั้น การดำเนินการดังกล่าวทำได้โดย

A x B = A0 + b0 + (A0 x B1 =A1 x B0) x Zw + (A1 x B1) x Z2u

A0 x B0 และ A0 x B1ถูกดำเนินการในระหว่าง วงรอบแรกของการคูณ และ A1 x B1 และ A1 x B1 เกิดระหว่างวงรอบที่ 2 ภายหลังผลวมทั้งหมดได้ถูกรวมใน 64-bit merge section ซึ่งเป็นส่วนประกอบที่สำคัญของ a set of array-saveadder trees (pipeline) ผลรวมและผลรวมที่เกิดจาก 64-bit merge-secction ได้ถูกเพิ่มโดยผู้บวกทั้ง 2 เพื่อนนำไปสร้าง 64-bit product

The star-100 มี 130 scalar instructions และ 65 vector instructions ดังเช่นที่ปรากฏอยู่ใน Table 4.1

Vectors ใน the star -100 ถูกสร้างเป็นแถวของเลขฐาน 2 หรือ ตัวอักษร หรือเป็น arrays ของ 32-bit หรือ 64-bit FLP numbers The sparse vector instructions สามารภสร้าง sparse vectors เมื่อ pipeline เริ่มทำการ streamng operations มันก็เป็นไปได้ที่จะทำด้วยอัตรา 40 ns ต่อไป เวลาที่ใช้ใน input --> output เพื่อเพิ่ม pipeline โดย FLP คือ 160 ns เพราะมันคือ 4 pipeline ที่สำคัญที่สุด เวลาของ FLP MULTIPLE PIPELINE ยืดออกไป = 320 ns มันเป็น CPU spreed สูงสุด , ในการดำเนินการ,ค่าความเร็วโดยเฉลี่ยของ star ประมาณ ถึง megaflops สำหรับ scalar operations และ 5 ถึง 15 megaflops สำหรับ vector operation, มันชัดเจนว่า double-precision FLP operation ต้องการเวลาเพื่อให้สมบูรณ์.

Texas Instrunent Advanced Scientific Computer(ASC) ได้ก่อกำเนิด ในปี 1972 . The control Processor ของ ASC ก็ได้รวมตัวกับ high degree of pipelining ใน instruction และ arimetic level. ส่วนประกอบพื้นฐานของระบบ ASC ดูได้จาก รูป 4.4 The central processing unit ถูกใช้กับ the operating system. ช่อง disk และช่อง tape รองรับตัวเลขเยอะๆ ของstorage unit . Data concentretors ได้ประกอบกันสำหรับรองรับแต่ละชุดและinteractive terminal ธนาคารความจำและหน่วยความจำหลักมีส่วนอิสระที่ทำงานเองได้ 8 ส่วน แต่ละส่วนใช้เวลา1รอบ 160 ns และ world lenght = 32 bit. 8 หน่วยความจำ สามารถส่งใน one memory . The memory control unit คือส่วนที่เชื่อมระหว่าง a memory buses กับ eight independent processor ports. แต่ละตัวดำเนินการ 8 ตัวดำเนินการอิสระ จะใช้ได้ง่ายที่จะหาหน่วยความจำอิสระทั้งหมด

The central processor unit (IPU) the memory buffer unit (MBU-Aupair) สามารถสร้าง ใน central processor ได้ The ASC instruction type ได้ถูกlist ไว้ใน table4.3ามเร็วสูงสุดของASC/ arimetic pipeline ก็กำหนด ให้ใน table4.4 เฉลี่ยอยู่ที่ประมาณ 0.6 ถึง1.5 magaflops และ 3 ถึง 10 megaflop ต่อ pipeline สามารถใช้ได้ทั้งsclar และ vector operation ตามลำดับ

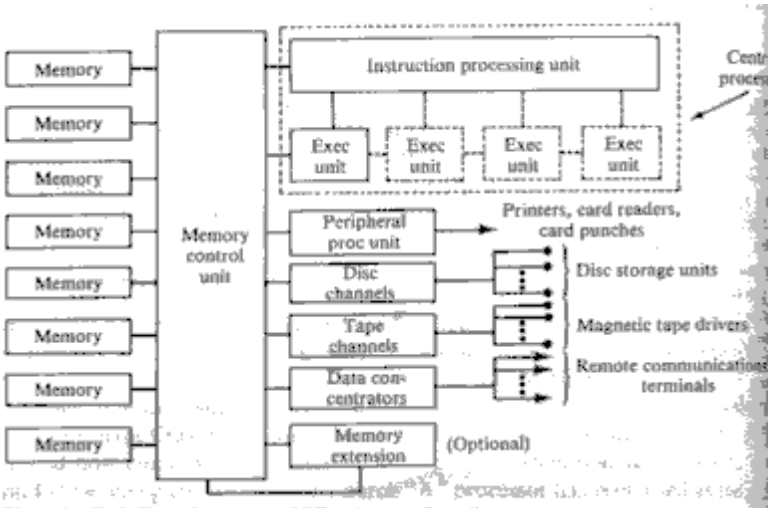


Figure 4.4 Basic Texas Instruments ASC systems configuration.

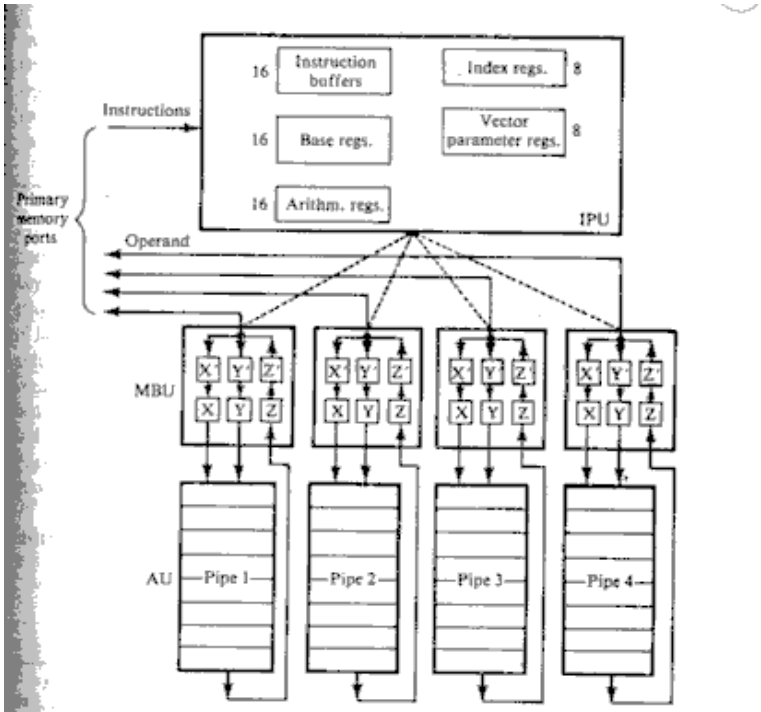


Figure 4.5 Central processor of the TI-ASC with four arithmetic logic pipelines. (Courtesy of Texas Instruments, Inc.)

The primary functions ของ IPU คือ การส่งส่วนที่เหลือของ(function ที่สำคัญที่สุด)

Central processor ด้อย continuous stream of instructions เกี่ยวกับภายใน the IPU คือ pipeline หลายส่วนที่มี 48 program address สำหรับดึงดูดและถอดรหัสและสร้าง operand address Instructions คือ ส่วนแรกใน octette (8 word) ดังนั้น IPU ดำเนินการโอน instruction ไปยัง MBU-AU ให้ได้รับสิ่งที่ดีที่สุดไปใช้ใน arithmetic pipeline The MBU เป็นตัวเชื่อมระหว่างหน่วยความจำหลักกับ arithmetic pipeline ฟังก์ชันที่สำคัญที่สุดของ

MBU คือการรองรับ arithmetic unit ด้วย continubus strems of operands The MBU มีตัวป้องกันโดยแต่ละตัวมี 8 ตัวบันทึก X และตัวป้องกัน Y ถูกใช้สำหรับ input และตัวป้องกัน Zถูกใช้สำหรับ output การนำมาและการเก็บรักษาข้อมูล ใช้ในรูป 8 word The AUมีโครงสร้างของ piptline to encble efficient erithmotic computations (ทำให้สามารถ,ผลกระทบ ,เกี่ยวกับตัวเลข,การคำนวณ) unitนี้ Ton ดังที่บรรยายใน section 3.2.2 ความคล้ายคลึงกันข้างล่าง The stcr-100 และระบบ ASC table 4.5 สรุปรูปแบบของ architectural Ton(การติดต่อกันระหว่างไมโครโปรเซสเซอร์กับระบบรอม)

2.2 Vector Procssing ใน Streamins Mode

continous streamins ของข้อมูลจาก The high bandwidth inter leaved memories ไปยัง mutiple pipeline นำไป stars มีผลกระทบอย่างมากในเวกเตอร์ยาวๆการออกคำสั่งนี้ไปยังโครงสร้างในการคำนวณในvector mode เช่น maxtrix multiplicetion ,polynomial evaluction และผลเฉลยเชิงเส้นของระบบสมการ ระบบของ star เตรียมการป้องกันปัญหาเพื่อให้ผู้ใช้ได้รับประโยชน์สูงสุดจากฮาร์ดแวร์ The Fortren compiler ใน star ได้ยืดออก Ton เส้นของรหัส Fortren ซึ่งสามารถ TON แน่นอนโปรแกรมสามารถหนีจากรหัส Fortren ไปใช้ภาษา Assemble โดยตรงเลยได้รูปแบบของ vector Instruction ของ star-100 อยู่ใน Fig 4.6 โดยแต่ละมี 68 bit ได้ถูกแบ่งเป็น 8 ส่วน ส่วน F และ G กำหนด function และsubfunction โดยในส่วนที่เหลือเป็นการกำหนดรูปแบบการทำงานในส่วน C+1 TON The effective staring address ถูกคำนวณจากผลรวมของ address พื้นฐานและหน่วยย่อยอื่นๆ The effective field lengh ถูกคำนวณจากหน่วยย่อยจาก field lengh ดังนั้น the ending address ผลรวมของ The effective staring address กับ

The effective field lengh เกี่ยวกับความสามารถของหน่วยย่อยส่วนประกอบ th bu แหล่งกำเนิด operand สามารถทำงานด้วยส่วนประกอบ (I +d)th ของแหล่งกำเนิด operand อื่นๆซึ่ง d คือความแตกต่างระหว่างส่วนประกอบ ตัวอย่างต่อไปนี้จะแสดง the streaming operation สำหรับ vector addition

The sterting address และ effective field length ของ vector A ถูกคำนวณใน Fig 4.7 ซึ่ง bit addressing ถูกใช้ และ a"l"ในcontrol vector ยินยอมที่จะเก็บ resulting vector ในส่วนที่เหมือนกันไว้ ยกตัวอย่างเช่น หน่วยความจำที่ 4000 ถูกเก็บในรูป a"l" ดังนั้น C5 ถูกแปลสภาพเป็น A5+B5 The skewing effect ถูกแสดงไว้ในตัวอย่าง หลังจากคำสั่งถูกแปลใน strecm unit แล้ว the eppropricte microcode sequence ก็เริ่มดำเนินการโดย the microcode unit (MIC) ใน stream unit

เมื่อ CPU เริ่มทำงานตามคำสั่งของ microcode control มันก็จะส่ง F(function) code และ a microcode ไปยัง MIC The Mic ครอบคลุมตามคำสั่งในกรณีของการยกเลิกคำสั่ง มันจะทำการ save operand ทั้งหมดและปัจจัยที่สำคัญทั้งหมดเพื่อกลับมาทำใหม่ในครั้งต่อไป The MIC คือหัวใจของ vector processing control ซึ่งจะทำงานตามขั้นตอนดังนี้

1. The reading of address จากแฟ้มบันทึก(ใน stream unit)สำหรับ vector parameter เพื่อกำหนดคำสั่ง

2.การคำนวณ the effective addresses และ field lenght สำหรับส่งสัญญาณของ vector operations

3.การกำหนด bs read – write bus ทำได้โดย G(sub – func) สำหรับ operand และ result

4.การแลกเปลี่ยนของ address และข่าวสารอื่นๆ เมื่อใดก็ตามจำเป็นต้องใช้ interrupt – count register ที่เหมาะสม

เมื่อ effective address ถูกคำนวณ , ส่วนประกอบของ operand ถูกนำมาและถูกแยกออกเป็นคู่ สำหรับ The operation involve โครงสร้างของ pipe จะทำงานจนกระทั่ง vector instruction สิ้นสุด จุดสิ้นสุดถูกกำหนดจากสิ่งต่อไปนี้ (c) c vector หมดเมื่อ effective field length กลายเป็น 0 ; (b)ข้อมูลบางส่วนหมดไป

ในการสนับสนุน vector และ scalar processing ใน star ระบบการทำงานของมันจัดแบ่งเวลา และใช้ให้เป็นประโยชน์จาก virtural memory TOM ระบบการทำงานของ star ควบคุมฟังก์ชัน input , compilation assembly , loading execution และ output program , TON ในส่วนของ star Fortres , as interactive interpreter ถูกเรียกว่า Star APL ถูกทำให้สำเร็จในระบบ , ซึ่ง upgrade ระบบ สามารถควบคุมพื้นนที่ใหญ่ๆๆของการคำนวณไว้

คำสั่งของ IT – ASC เป็น 32 bit , แสดงใน รูป 4.3 , ซึ่ง F คือ opcode , R ,T,และ M กำหนด arithmefic , index และ base register และ N คือ symbolic address . ASC แตกต่างจาก star ในรูปแบบของคำสั่ง vector แทนที่การบันทึกไปยัง operand address และ control information , the ASC ใช้ vector parameter file (VPF) , ซึ่งประกอบด้วยตังบันทึก 8 ตัว 32 bit ใน IPU แสดงในรูป 4.6 ฟังก์ชันของแต่ละตัวบันทึก ถูกกำหนด ในรูป 9.3 ตัวบันทึก V0 holds the opcode , the vector operand type และ the length ; V1 ,V2 และ V3 แนะนำ base address และ การแทนที่ของแต่ละ operand vector ; V4 และ V5 holds การเพิ่มขึ้นของ vector index และตัวเลขของ inter loops ; V0 และ V3 holds ข่าวสารที่เหมือนกัน สำหรับ outer loops

3. Sciencetific Attached Processor

Attached Prcessor เป็นที่นิยมเพราะว้าราคาถูกและกระนั้นได้มีการเตรียมการแก้ไขปรับปรุง บนเครื่อง ซึ่ง AP-120B และ FPS-164 เป็น back-end มีการออกแบบหน่วยประมวลผลการคำนวณพิเศษ ถึงขั้นตอนขนาดใหญ่ของ Vector หรือ Arrays(Matrices) ของข้อมูล คำสั่งนี้สามารถทำการประมงลผลได้จำนวนมาก เกี่ยวกับ host computer ซึ่งสามารถทำระหว่าง minicomputer (เช่น VAX-11 series) หรือ Mainfraim computer (IBM 308 x series) ขณะที่ host computer ควบคุมระบบทั้งหมด และดูแล I/O และอุปกรณ์ตอบนอก Attacted Processor สามารถตอบสนองสำหรับ การคำนวณตังเลขที่มีจุดทศนิยมมาก ๆ เช่นมันสามารถกระจายหน้าที่หาผลลัพธ์ ในเวลา 200 หน่วยเวลา ในเครื่อง Minicomputer และใช้เวลา 20 หน่วยเวลา สำหรับเครื่อง Mainfraim , Attached Processor อื่น ๆ ที่ทำงานทางวิทยาศาสตร์ ได้แก่ IBM 3838 และ Datawest Processor ซึ่งมีราคาถูกด้วย ลักษณะเด่นทางสถาปัตยกรรมของ Attached Procesor คือความสามารถ และประโยชน์ในการทำงานทางด้านวิทยาศาสตร์ และ วิศวกรรม

.

3.1 สถาปัตยกรรม ของ AP-120B

การทำงานร่วมกันของ AP-120B และ host computer แสดงอยู่ในรูป 4.9 อุปกรณ์ภายนอกทั้งหมดได้แก่ printers , display monitor , disk และ tape units จะทำงานร่วมกับ host computer ที่จริงแล้ว AP-120B จะเป็นตัวควบคุมการทำงานของอุปกรณ์ภายนอกแล้วค่อยจัดการทำงานกับ host ที่ต้องการใช้งาน เนื่องด้วย host และ back-end อาจมีรูปแบบข้อมูลต่างกัน และความยาวของ word ไม่เท่ากันได้ interface unit ต้องการเปลี่ยนแปลงข้อมูล " on fly " และมีเครื่องมือช่วยสนับสนุน direc-memory access (DMA) และ programmed input-output (PIO)ในการขนส่งข้อมูล มีการจัด register ออกเป็น 2 ส่วน ใน interface unit กลุ่ม 1 จะควบคุม function ผ่านทางโปรแกรม I/O ของ interface unit เตรียม array processor กับ simulate front panel ของ host โดยจะบรรจุ switches register ใช้โดย host เข้าไปควบคุม หรือ ข้อมูลพารามิเตอร์ และ

Addresses ใน array processor light register แสดงรายละเอียดใน array processor และ function register สำหรับแสดงคำสั่ง เช่น start , stop , reset

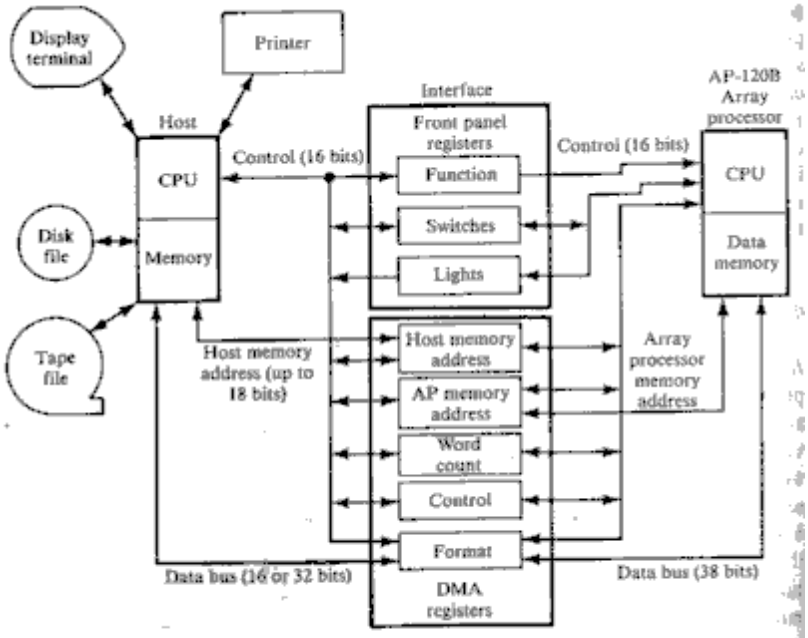


Figure 4.9 The AP-120B host and interface organization.

DMA register set ประกอบด้วย host memory address register AP memory register , word count register , control register , format register จะเป็นตังบังคับทิศทางการขนส่งข้อมูล และ

mode ของการขนส่งข้อมูล format register จะทำการแปลง ระหว่าง FLP format ของ host และของ AP-120B Interface logic จะอนุญาตการส่งผ่านข้อมูลภายใต้ การควบคุมของแต่ละ host หรือ AP-120B รูปแบบของเลขจุดทศนิยมใน AP-120B ใช้ 38 bits โดยเป็น 2 ‘s complement 28 bits และเลขยกกำลัง 10 bits biased by 512 มีการแก้ไขและได้ข้อตกลงที่แน่นอน ของรูปแบบเลขจุดทศนิยมเป็น 32 bits ถ้า host มีข้อมูลเลขจุดทศนิยมแตกต่างไป รูปแบบนั้นจะถูกทำการแปลงด้วย “ on the fly “ ผ่าน interface ดังนั้น AP-120B ทำงานหนักเกี่ยวกับการคำนวนได้เต็มที่

รูปแบบแปลน function ของ AP-120B แสดงอยู่ในรูป 4.10 processor แบ่งการทำงานได้ 6 ส่วน คือ I/O section , memory section , control memory , control unit , data bus และ arithmetic 2 ส่วน ในส่วนของ memory ประกอบด้วย data memory (MD) , table memory (TM) และ 2 data pads (DPX and DPY) control memory หรือ program memory มีขนาด 64 bit words กับ 50 ns cycle time program memory ประกอบด้วย 4 K words in 256 word Instruction register เป็น fetch , decoded และ executed ปนส่วน control unit datamemory ใช้เวลาระหว่าง 167 หรือ 333 ns ความเร็วนี้ขึ้นอยู่กับ trade-off ระหว่าง ราคากับประสิทธิภาพ ตัว data memory ใช้เนื้อที่ใน main data storage 38-bit words data memory addressable ได้โดยตรง โดย 1 ล้าน words ใช้เวลา 167 หรือ 333 ns TM มีส่วนร่วมกับ special data path ซึ่งแต่ละส่วนจะไม่ก้าวก่ายกับ data path ที่ทำงานร่วมกับ data memory data pads X และ Y เป็น 2 block ของ 38-bit ซึ่งแบ่งเป็น 16 accumulator แต่ละ block accumulator จะ addressable ได้โดยตรงผ่าน AP processor ทุก accumulator สามารถ access ใน single machine ด้วยเวลา 167 ns การอ่าน และ เขียนแต่ละสมการสามารถทำได้ในเวลาเดียวกัน

S pad ใน control unit มี 2 ส่วน คือ S-pad memory และ integer ALU S-pad memory ประกอบด้วย 16 directly address ตัว register นี้ใส่ address ALU ทำให้การ operand address ได้ผลดี ตัว address ALU จะทำการคำนวณด้วยเลขจำนวนเต็ม 16 bit ผลลัพธ์ที่ได้จาก address ALU ทำตาม ตัว register , MA สำหรับ data memory , TMA สำหรับ table memory และ DPA สำหรับ data pads และฟังก์ชัน อื่น ๆ ของ address ALU รวมทั้ง clear , increment , decrement , logical , and logicator

2 pipeline arithmetic units เป็น FLP adder (FA)และ FLP multiplier (FM) FA ประกอบด้วย 2 input register คือ A1 และ A2 และ two-segment pipeline ซึ่งแสดงในรูป 4.11 ผลรวมของ output คือ 38 bit เป็นเลขจุดทศนิยม ตัว FM มี M1 และ M2 input register และ three-segment pipeline ซึ่งจะกระทำการคูณเลขจุดทศนิยม pipeline เต็ม ผลลัพธ์ใหม่ที่ได้ของทุก machine cycle คือ 167 ns เนื่องจาก maximum through put rate ของเครื่อง AP-120B เป็น 12 mega floating-point ต่อวินาที เคื่อง AP-120B สามารถคำนวณได้ประสิทธิภาพสูง จาก sections ของ processor ที่มีหลายตัว โดยมันใช้ 2 pipeline arithmetic unit (FA and FM) 1 integer ALU ,

memory หลายตัว (PM,MD,TM) ซึ่งทำงานเป็นอิสระต่อกัน ตัวเลขที่มีขนาดใหญ่ของ register และ accumulators (A1,A2,M1,MA,TMA,DPA,DPX และ DPY) และ seven data path แสดงโครงสร้างของ bus ในรูป 4.10

two floating-point arithmetic units คือ FA และ FM มันใช้ 2 block ที่เป็นอิสระ ของ accumulator (DPX ,DPY) จะจัดเตรียมเปลี่ยนแปลงใน inhandling operands และ intermediate และผลลัพธ์สุดท้าย ตัวอย่างแต่ละ block สามารถบรรจุ vector operand กับ 16 components โดยสามารถกระทำข้างใน FA และ FM ได้ ขณะที่ block อื่น ทำการรับส่งข้อมูล จาก data memory หรือ table memory

โครงสร้าง pipeline ของ FA และ FM เป็นลักษณะ below stage แรกของ FA คือการเปรียบเทียบ ค่ายกกำลังเพื่อเป็นตัวเลขขนาดเล็ก ใน stage ที่สองผลลัพธ์เป็น normalized และ rounded เพราะความเร็วของ processing ต่างกัน ใน 2 stage นี้ buffer จะบรรจุเข้าไปในระหว่างหาผลลัพธ์ output ของ F:P adder โดย FA สามารถมี จุดมุ่งหมายได้แตกต่าง 5 เส้นทาง Possible source ติดต่อกับ input regiter A1 และ A2 ดังแสดงในรูป 4.11 ตัว FM มี 3 stage โดย stage แรกใช้ 56 bit โดยแบ่งเป็น 2 ส่วน ๆ ละ 28 bit stage ที่ 2 complete the product of the fraction stage ที่ 3 บวกเลขยกกำลัง , rounds และ normalizes the fraction possible source ทั้งหมด เชื่อมต่อเข้ากับ FLP multiplier ดังแสดงในรูป 4.12

Seven buses ใน AP-120B ทำงานได้พร้อมกันทำงานให้ทำงานแบบ parallel processing ได้ FA และ FM มี multibus input ports ทั้งคู่ ใน words อื่น ๆ operands หลาย ๆ ตัว และผลลัพธ์สามารถ moved ระหว่าง different function unit ใน machine cycle เดียวกัน total data path banwidth จะ math กับ execution speed ของ pipeline adder และ multiplier

หลาย ๆ levels ของ pipeline ใน AP-120B จะ มี described parallel function units จะจัดเตรียม long instruction word ของ AP-120B และมี 64 bits ซึ่งแบ่งย่อย ใน 10 command fields (รูป 4.13) แต่ละ command field จะควบคุม specific unit เป็น single AP-120B instruction สามารถเริ่มได้หลาย 10 operations ต่อ machine cycle อยู่ในรูป 4.13 multiple memory access register transfer , interger arithmetic และ floating-point computations สามารถเกิดขึ้นในเวลาเดียวกัน

ในผลรวม multiple memories multiple functional units parallel data path และ multiple command field ใน คำสั่งทำโดย attached processor ความเร็วสูงสำหรับงานทางวิทยาศาสตร์

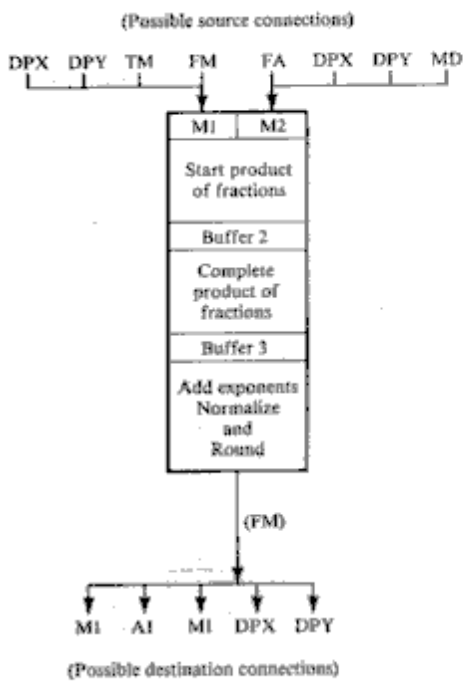


Figure 4.12 The floating-point multiplier in AP-120B.

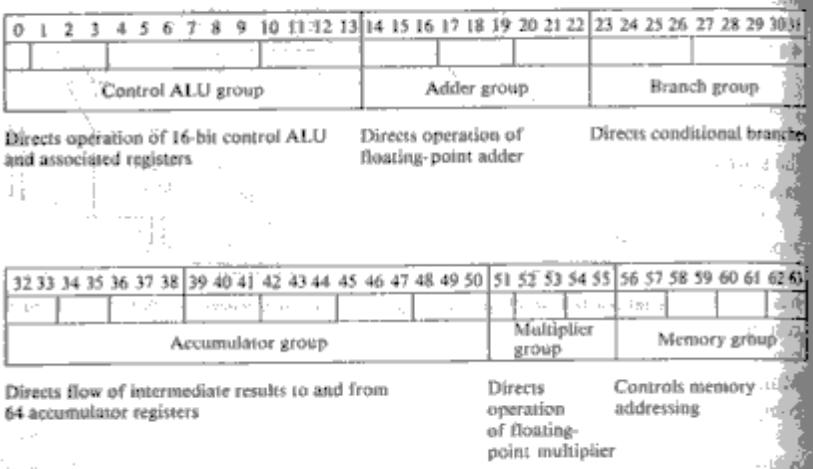


Figure 4.13 The instruction format in AP-120B.

3.2Back-End Vector Computation

เครื่อง AP-120B ไม่เหมือนกับ vector supercomputers มันไม่มี vector instruction แทนที long instruction บรรจุ microoperations ใช้เฉพาะส่วน parallel software มากกว่า 200 packages มีการพัฒนาที่ซับซ้อน ในการคำนวณแบบ vector สิ่งนี้เราเรียกว่า mathematical libraly vector-processing routine เป็น Fortran callable จาก host computer ทั้งหมด calls are handled โดย array processor executive (APEX) soft ware ขณะ decodes Subroutine เรียกจาก Fortran programs อยู่ใน host และ automatically passes ควบคุม parameter ไปยัง AP-120B จะ return ผลลัพธ์ ไปยัง host computer ที่ต้องการ ผู้ใช้สามารถ add new หรือ special routines ไปยัง mathematics libraly ใน Fortran หรือ ใน AP assembly language code บาง program พัฒนาขค้นมาใช้งานโดยเฉพาะ In addition เป็น signal processing

libraly dedicated ไปที่ dagital signal processing application ที่สำคัญ Fortran callable routines ใน libraries นี้ ดังตาราง 4.6 กับ time measured ใน microsecond และ program size ใน numbers ของ AP-120B microinstruction words

ใน AP-120B มีฟังก์ชันที่แตกต่าง ทำงานโดย floating-point adder ในเวลาที่แตกต่างกัน รายละเอียดข้างล่างเป็นประเภท ของ ฟังก์ชัน บางอย่าง

1. A1 + A2
2. A1 – A2
3. A2 – A1
4. A1 EQV A2
5. A1 AND A2
6. A or A2
7. Convert A2 from signed magnitude to 2's complement format
8. Convert A2 from 2's complement to signed magnitude format
9. Scale A2
10. Absolute value of A2
11. Fix A2

Table 4.6 Floating-point arithmetic timing for some functions in AP-120B

Operation	Travel time	Pipeline interval
Add-subtract	333 ns	167 ns
Multiply	500 ns	167 ns
Multiply-Add	833 ns	167 ns
Complex add-subtract	500 ns	333 ns
Complex multiply	1333 ns	667 ns
Complex multiply-add	1667 ns	667 ns

Table 4.7 Important Fortran callable routines for AP-120B

Operation	Name	Timing (us per point)	Size (AP-120B prog. Words)
Real vector operations			
Vector add	VADD	1.2	8
Vector subtract	VSUB	1.2	8
Vector multiply	VMUL	1.2	11
Vector divide	VDIV	1.8	44
Vector exponential	VEXP	5.1	42
Vector sine	VSIN	5.1	46
Vector cosine	VCOS	5.6	46
Sum of vector squares	SVR	0.4	11
Dot Product of two vectors	DOPTR	0.8	9
Sum of vector elements	SVE	0.4	7
Complex vector operations			
Complex vector multiply	CVMUL	2.0	26
Complex vector reciprocal	CVRCIP	5.0	51
Matrix operations			
Matrix transpose	MTRANS	0.8	17
Matrix multiply	MMUL	*	58

Operation	Name	Timing (us per point)	Size (AP-120B prog. Words)
Matrix multiply (dimension <= 32)	MMUL32	*	27
Matrix inverse	MATINV	*	130
Matrix vector multiply (3 x 3)	MVML3	2.5/vector	30
Matrix vector multiply (4 x 4)	MVML4	4.6/vector	39
Fast fourier transform operations			
Complex FFT	CFFT	*	187
Real FFT	RFFT	*	235
Signal processing operations			
Convolution(or correlation)	CONV	*	102
Wiener-Levinson algorithm	WIENER	*	68
Bandpass filter	BNDPS	*	287
Power spectrum	PWRSPC	*	268

* Timig unknown.

ในลักษณะคล้ายคลึง FM ที่สามารถทำงานได้หลายฟังก์ชันที่แตกต่างกันได้ เวลาในการคำนวณ floating-point ใน AP-120B มีผลดังตารางที่ 4.7 ที่ travel time เป็นเวลารวมที่ใช้ในการ transfer ข้อมูลจาก source ไปยัง destination และ pipeline interval เป็นเวลา ระหว่าง successively avaliable result pipline interval บ่งบอก maximum throughput rate สำหรับ vector oriented computations

ตัวอย่างรายละเอียดของ vector processing in the AP-120B ให้อยู่ด้านล่าง เครื่องหมาย semicolon “ ; ” จะแยก parallel operations ภายในคำสั่ง comma “ , “ เป็นตัวแยก operands double slash “ // “ เป็นส่วน comment arrow “←” อ้างถึง การแทน operator สำหรับการ รับส่งข้อมูล บาง operations แสดงตัวอย่างเฉพาะทางด้านล่าง

Fadd A1 , A2	// A1+A2 (floating-point add)
DPX(m) ← FA	// Save FA in location m of data pad DPX
FMUF M1 , M2	// M1 x M2 (floating-point multiply)
DPY(m) ← FM	// Save FM in location m of data pad DPY

ลำดับการคำนวณ cycles ที่ 1 ถึง 3 ใช้ FM pipeline เต็มที่ cycles ที่ 4 ถึง 5 จะใช้ FA pipeline เต็มที่ cycles ที่ 6 ถึง 8 ใช้ FM pipeline สิ้นเปลืองมาก cycles ที่ 9 ถึง 11 FA pipeline จะถูกใช้อย่างสิ้นเปลือง และผลลัพธ์สุดท้ายถูก store ใน data pad x The dummy add " FADD " ปราศจากการโต้ตอบใน cycles ที่ 11 ซึ่ง ใช้เฉพาะ last computation out of pipeline 3 stage ที่กล่าวมาแล้ว และ 2 buffer register ใน FM pipeline เพราะฉะนั้น 2 dummy multiplier ต้องการ push the last two computations out of the pipeline สำหรับ long vector ความเร็วในการ execute dot product ใน AP-120B มีความเร็วมากใน serial processor

AP-120B มีการประยุกต์ extensively ใน field ของ digital-signal processing การ execute ตามลำดับ ของ fast Fourier transform (FFT) ใน AP-120B มีตัวอย่างแสดง ด้านล่าง FFT program อาศัย program memory ของ AP-120B array ของข้อมูลจะ transformed ใน main memory ของ host computer FFT compution มีลำดับตาม step ดังนี้

1. The host computer issues an I/O instruction to initiate the FFT program in the AP-120B
2. The AP-120B request host DMA cycles to transfer the array of data from host memory to data memory in the AP-120B. the floating-point format is converted during the flow of data through the interface unit
3. The FFT computations are performed over a 38-bit floating-point dataarray .
4. The AP-120B requests the host DMA cycles to return the results of the FFT frequency-domain coefficients array.

FPS-164, IBM 3838 AND DATAWEST MATP

The FPS-164 เป็นวิวัฒนาการมาจากสถาปัตยกรรมของมันเอง The ap-120B, The AD-190L และ The FPS-100 โดยระบบจุดทศนิยม FPS-164 ได้มีส่วนประกอบเข้ามาในส่วนของช่อง input-output หรือช่อง DMA ของเครื่องคอมพิวเตอร์แม่ข่ายโดยวิธีทางฮาดร์แวร์และตัวเชื่อมต่อที่เหมือนกับซอฟท์แวร์นั้นสำหรับ The ap-120B เครื่องจักร

DEC VAX 11/780, IBM 4341 และ IBM 3081 กำลังขยายเป็นวงแหวนจากซุปเปอร์มินิไปยังแมนเฟรมที่มีขนาดใหญ่ The FPS-164 มีประสิทธิภาพที่ดีกว่า ap-120B โดยทำให้มีความแมนยำมากขึ้น (ระบบเลขทศนิยมมาจาก 38 บิต เป็น 64 บิต) และขยายหน่วยความจำ 16 ล้าน 64 บิต FPS-164 สามารถโปรแกรมนอกจาก Fortran-77 ภาษาสัญลักษณ์
FPS-164 หรือการขยายไลบารี่ของโปรแกรมทางคณิตศาสตร์ เมทริกซ์ และการประยุกต์ที่ใช้ประจำวัน

ฟังก์ชันผังกรอบภาพของ FPS-164 คือ ทำให้ใน รูป 4.14 สิ่งเหล่านี้เป็น 8 หน่วยฟังก์ชัน pipeline อิสระ(ตัวคูณ the FPS, ตัวบวก the FPS ในชุดของข้อมูล X และ Y ,ตรวจหน่วยความจำ, ตัวเลขจำนวนเต็ม ALU และข้อมูลชุดเส้นทาง) เชื่อมต่อโดย 7 เส้นทางเฉพาะข้อมูลความเร็วสูงสุดยังคงคือ 12 เมกะฟลอพ ข้อมูล 64 บิตจัดเตรียมไว้สำหรับ 15 ตัวเลขฐานสิบที่แน่นอน แอดเดรสที่ว่างของ 64 บิตครอบคลุมถึง 16 ล้านคำการป้องกันของผู้ใช้หลายคนทำโดยใช้เนื้อที่ฐานและรีจิสเตอร์และคำสั่งเอกสิทธิ์
เวกเตอร์ลำดับก่อนหลังขัดจังหวะจริงในการประยุกต์ แนวที่ไม่คงที่และแน่นอนของ The FPS-164 ปรับปรุงเครื่องหมายมากกว่า AP-120B ยิ่งกว่านั้นโปรเซสเซอร์มีชุดคำสั่งซึ่งเป็นเครื่องมือช่วยเหลือซอฟท์แวร์ของดับเบิ้ลจุดทศนิยม การวินิจฉัยและความเชื่อถือได้มักจะสร้างไปใน FPS-164 ทำให้ได้ผลที่ชัดเจนของระบบขึ้นอยู่กับกรณีที่ฮาร์ดแวร์หรือซอฟท์แวร์ที่ใช้งานไม่ได้

The IBM 3838 คือ mutible- pipeline โปรเซสเซอร์พิเศษ มันพัฒนามาจาก IBM 2938 ซึ่งโปรเซสเซอร์ทั้งสองตัวถูกสร้างขึ้นมารองรับ IBM เมนเฟรมโดยเฉพาะเหมือนกับระบบ 370 ที่ทำงานเพื่อรองรับความสามารถในการทำงานของ vector ของเครื่องจักรแม่ข่าย เหล่านี้เป็นตัวช่วย pipeline โปรเซสเซอร์ให้ยังคงประมวลผลต่อไปใน IBM รุ่นถัดไปคือ 360/91 และ 370/195 ชุดคำสั่งของ vector มันสามารถกระทำการใน 3838 ประกอบไปด้วยชิ้นส่วนการบวกเวกเตอร์ ,การคูณเวกเตอร์,ข้างในผลผลิต,ผลรวมของชิ้นส่วนการบวกเวกเตอร์,การเคลื่อนเวกเตอร์ เหมือนกับ AP-120B และ FPS-164
ทั้ง AP-120B และ FPS-164 ถูกโปรแกรมภาษาเครื่องโปรเซสเซอร์ pipeline ที่ซึ่งสามารถรองรับกับชุดคำสั่งเฉพาะสำหรับการประยุกต์เวกเตอร์

โครงสร้างฮาร์ดแวร์ของ IBM 3838 ชุดโปรเซสเซอร์จะแสดงใน รูป 4.15
โปรเซสเซอร์สามารถรองรับระบบ 370 โดยผ่านทางช่องผสมสัญญาณ I/O กับข้อมูลที่ส่งไป 1.5 MB/S กับ 2 ทางเลือกของประเภทตัวเชื่อมต่อ คือ การส่งข้อมูลที่มาก
สามารถเพิ่มเป็น 2 เท่าคือ 3 MB/S 3838 จะปรากฏในช่องโปรเซสเซอร์ I/O เช่น แบ่งส่วน control unit ผู้ใช้ 7 คนสามารถกระทำพร้อมๆกันได้ใน 3838 การทำงานถูกกำหนดโดย pipeline ต่างๆในระบบย่อยที่อยู่ใน 3838 ตัวควบคุมโปรเซสเซอร์สามารถช่วยเหลือผู้ใช้กับชุดของคำสั่งและจำเป็นต้องไปเตรียมชุดคำสั่งเวกเตอร์ หน่วยความจำขนาดใหญ่ที่เคยเก็บตัวคูณเวกเตอร์ หน่วย I/O ดูแล

การส่งผ่านข้อมูลหรือโปรแกรมระหว่างโฮส และเมมโมรีขนาดใหญ่ ขนาดของข้อมูล 3838 คือ 32 บิต ระบบที่ใช้คือ 370

การขนส่งของการทำงานหนังสือชุดของส่วนของเวกเตอร์ระหว่างขนาดเมมโมรี่ และการเก็บงานที่ทำซึ่งถูกดูแลโดยเวกเตอร์ datd transfer controller (DTC) แต่ละการทำงานสามารถเก็บได้ 8192 ไบท์ แอดเดรสเวกเตอร์จะถูกจัดส่งไป DTC สามารถกระทำแปดรูปแบบข้อมูลในระหว่างที่กำลังไหล arithmetic control มันจะเป็นหน่วยโปรแกรมภาษาเครื่อง ซึ่งโปรแกรมภาษาเครื่องนี้มักจะเป็นส่วนหนึ่งที่ถูกกระทำโดยตัวเลขของ pipeline ที่ถูกกำหนดโดยตัวควบคุมตัวนี้ DTC การใช้การทำงานเก็บโดยตัวเลขของ pipeline และโดย DTC ที่นับจังหวะได้โดยพื้นฐานเวลาต่อรอบคือ 100 ns ในเครื่อง 3838 นั่นเป็น 5 หน่วยตัวเลข pipeline ใน 3838 หน่วย pipeline เหมาะกับไดอะแกรมใน รูปท่ 4.15 ประกอบด้วย 2 ที่อยู่เลขทศนิยมของ แต่ละสถานะ 4 สถานะเลขทศนิยมตัวคูณ, 5 สถานะ sine/cose pipeline และ 5 สถานะการแลกเปลี่ยนซึ่งกันและกัน แต่ละการทำงานเก็บอย่างเช่น 4 สถานะ pipeline

อย่างช้าจะทำงาน 100 ns ในแต่ละสถานะเส้นทางการเชื่อมต่อระหว่างฟังก์ชัน pipeline นี้จะอยู่ภายใต้ การควบคุมของโปรแกรมภาษาเครื่องของ arithmetic การดึงของความสามารถในการเขียนควบคุมแต่ละสถานะมักจะเป็น pipeline กับ 2 สถานะที่ช้า

โปรแกรมและข้อมูลที่ถูกประมวลผลโดย 3832 จะถูกส่งผ่านโดย host ของคอมพิวเตอร์ทั้ง vector และคำสั่ง scalar สามารถบรรจุในโปรแกรม 3838 สั่งโปรแกรมและข้อมูลไปช่อง I/O ข้อมูลจะถูกเก็บในที่เก็บขนาดใหญ่ คำสั่งจะถูกการะทำการโดย control processor หลังจากแปลแต่ละคำสั่ง control processor จัดเตรียมเส้นเชื่อม ส่วนหนึ่งของโปรแกรมสำหรับดูแลการกระทำ pipeline ของแต่ละคำสั่งอย่างไรก็ดีตัวเลข pipeline จะอัพเกรดข้อมูลเวกเตอร์จากหนึ่งการเก็บทำงานองค์ประกอบของ mutiprogram ใช้ของ 3838 ความเร็วสูงสุดของ 3838 ประมาณ 30 เมกะฟลอพ

Datawest ที่ Scohtsdale , Arizona สร้างให้รองรับ processor ที่เรียกว่า MATP สำหรับการคำนวณเฉพาะทางวิทยาศาสตร์ MATP ประกอบจาก 4 pipeline โปรเซสเซอร์ซึ่งเป็นรูปแบบการผสมกันของ MIMD-SIMD ซึ่งสามารถโปรแกรมภาษาเครื่องและแบ่งรวมข้อมูลหน่วยความจำโปรเซสเซอร์แต่ละตัวสามารถควบคุมโดยการแบ่งตามการใช้เขียนควบคุมการเก็บ เมนหลักของแม่ข่ายการติดต่อคือเชตของช่องโปรแกรมนั้นติดต่อกับช่องของ I/O เปรียบเทียบทั้ง 3 ตัวที่สนับสนุนโปรเซสเซอร์ผลิตใน United States จำใช้ในตาราง 4.8 โปรเซสเซอร์ 3 ตัว FPS’AP-120B, IBM’s3838 และ Datawest’s MATP เป็น pipeline และการโปรแกรมภาษาเครื่องความเร็วของ MATP มีลักษณะอย่างเดียวกับ 4 processor มันเป็นสิ่งที่น่าสนใจในโครงสร้างของ MATP ตัวสนับสนุนชุดของ Processor มีผลต่อ scismic-signal

4 Vector Processor in Cyber-205 and CDF-NASF

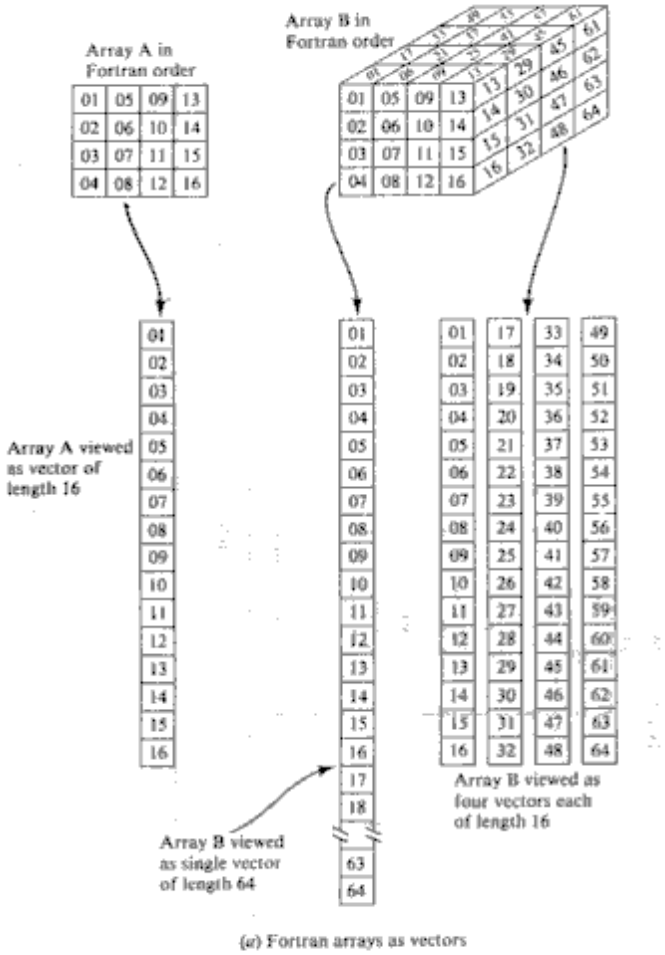
ใน section นี้เราจะอธิบายถึงลักษณะพิเศษหลายๆลักษณะ ใน Cyber-205 ลักษณะเหล่านี้เป็นสิ่งสำคัญที่ทำให้ง่ายต่อการจำ vector pipeline อย่างแรกเราจะต้องทบทวนระบบ memory-mapping ใน Cyber-205 เมื่อเราแสดงการใช้ประโยชน์ของ bit vectors สำหรับควบคุม vector processing ผลที่ทำให้ Cyber-205 ทำงานช้าจะถูกประเมิน เราจะอธิบายการปรับปรุง I/O configuration ใน Cyber-205 ซึ่งเปรียบเทียบกับระบบ Star-100 ตอนเริ่มแรก สุดท้ายเราจะได้เรียนรูปแบบการเพิ่มการประมวลผล โดย Control data

The Cyber-205 ถูกจำกัดการทำงานด้วยการทำงานที่รวดเร็วของ memory-to-memory โดยพิจารณาจาก Fortran declaration DIMENSION A(4,4) , B(4,4,4) ในการทำ memory- allocation ของFortran array A จะถูกพิจารณาโดย vector ความยาว 16 และ vector B มีความยาว 64 ถ้าข้อมูลทั้งหมดถูกทำให้เป็นเหมือนหน่วยของส่วนประกอบอิสระ ทั้งหมดของประเภทอื่น ถ้าทุกๆแผนกอื่นๆของ array B จะถูก process เราสามารถดูข้อมูลโดยใช้ vector ทั้ง4ได้ข้อมูลที่ถูกเก็บจะอยู่ติดๆกันใน memory ในทางวิศวกรรมมีทางแก้ไขโดยใช้แถบสายที่กว้างและเร็วขึ้น แต่แนวคิดของการแก้ปัญหาในรูปแบบของ vector ถูกนำมาใช้ประโยชน์โดยนักคณิตศาสตร์ ในการพัฒนาขั้นตอนการคำนวนแบบใหม่

แนวคิดของ memory-to-memory vector pipelining ใน Cyber-205 จะแสดงให้เห็นในรูปภาพที่ 4.28 ตั้งแต่การมาถึงของข้อมูล ที่ arithmetic pipeline ซึ่งปกติจะติดกัน ทุกๆส่วนใน pipeline จะแสดงการทำงานที่เป็นประโยชน์เว้นแต่ที่การเริ่มต้น และการสิ้นสุดที่มาของการจัดการของ vector จากมุมมองทางวิศวกรรม กระบวนการนี้ทำให้การทำงานของวงจรไฟฟ้ามีประสิทธิภาพ

จากประสบการณ์ที่แสดงว่า supercomputers จำนวนน้อยจะจัดการกับปัญหาใหญ่ปัญหาเดียวในการจัดการ การติดตั้งชั่วโมงของการปลุกจะถูกใช้ไปจนหมดในการ พัฒนา algorithm ,การแก้ debug

ของโปรแกรม และการปฏิบัติงานที่ทำปฏิกริยาของโปรแกรมวิจัย และโปรแกรมการผลิตขนาดใหญ่ ชั่วโมงในตอนเย็นและเที่ยงคืนปกติจะถูกสร้างขึ้นมาด้วยหนึ่งโปรแกรมหลักที่ทรัพยากรของเครื่องกลหรือมากกว่า ด้วยจำนวนที่พอประมาณของผลสะท้อนต่อการเสาะหาและจำกัดข้อผิดพลาดซึ่งจะยังคงดำเนินต่อไปใน background mode.



(a) Fortran arrays as vectors

Figure 4.27 Memory mapping examples in the Cyber-205.

Virtual memory จะแสดงบทบาทหลักในการสนับสนุน supercomputer รายการข้างล่างนี้เป็นสิ่งสำคัญของ memory ใน Cyber-205

1.การช่วยเหลือในการจัดการความจำ : ระบบการจัดการสามารถแสดง block ที่กำหนดเอาเองอย่างเปิดเผยของความจำจริง โดยปราศจากการรับรองการติดกันทางกายภาพของผู้ใช้ workspace การกระทำที่เพิ่มจนท่วมหัวนี้ เช่น การเพิ่มขึ้นของช่องว่างที่ไม่ได้ใช้ ซึ่งมีราคาแพงใน memory ขนาดใหญ่(บนคำสั่ง 8,000,000 คำสั่ง)

2.การปฏิบัติงานที่ปรากฏให้เหมือนกันทุกประการของทุกๆงาน :การปฏิบัติงานนี้หมายถึงโปรแกรม dimension statement และการใส่ parameter เข้าไปยังคงไม่เปลี่ยนแปลงอันหนึ่งอันใด

หรือไม่ 4-hour run กำลังถูกพิจารณาหรือการเสาะแสวงหาและกำจัดข้อผิดพลาดง่ายๆของขั้นตอนที่มุ่งหมายไว้โดยเฉพาะ

3.การกำจัดช่องว่างจากการพัฒนา algorithm : นักคณิตศาสตร์ และโปรแกรมเมอร์ ยังคงสามารถพัฒนา algorithm ซึ่งถ้าพวกเขาเคยหา workspace ที่ไม่มีขอบเขตในการใส่ข้อมูลและผลลัพธ์ชั่วคราว ครั้งหนึ่งที่ algorithmถูกพัฒนา โปรแกรมเมอร์ต้องเสนอแนะความหมาย เพื่อจัดการหน้าของข้อมูล เพื่อทำให้การเสนอของระบบมีประสิทธิภาพ และเพื่อกำจัดการทำกลับไปกลับมาซึ่งสามารถเกิดขึ้นใน virtual memory machines ในการเคลื่อนย้ายข้อมูลไปให้และ จาก real central memory.

ส่วนใหญ่การให้ความหมายของ Cyber-205 เป็นความคิดที่เป็นสายของ binary bits เรียกว่า (bit vectors) สามารถนำข้อมูลเกี่ยวกับ vectorsและสามารถประยุกต์ vectorsเหล่านั้น เพื่อแสดงหน้าที่สำคัญบางหน้าที่ นับตั้งแต่ bit strings กลายมาเป็นสิ่งสำคัญสำหรับแนวคิดการสร้างใหม่ของ vectors วิธีการต้องถูกเตรียมในการเปลี่ยนแปลง bit vector ให้เหมาะสมให้ดีเหมือนกับ numeric vectors ดังนั้น การเพิ่ม string function unit เข้าไปใน hardware ensemble ทั้งใน Star-100 และ Cyber-205 หน้าที่ของ compress mask , merge , scatter และ gather ถูกนำมารวมทั้งหมด ในอีกด้านหนึ่ง ส่วนมากของขั้นตอนการลดลงเหมือนกับ sum ,product และ inner product ซึ่งดำเนินการโดยตรงใน hardware

คำสั่งพิเศษ 2 คำสั่งของ vector ใช้ในการควบคุม bit vectors ซึ่งอธิบายการยกตัวอย่างในรูปที่ 4.29 ในส่วน a แหล่งกำเนิด vector 2 แหล่งA และ B ถูกทำให้รวมกันภายใต้การควบคุมของ bit vectors C ซึ่งให้ผลลัพธ์ของ vector R การทำให้รวมกัน นำมาเลือกจาก Aที่อยู่ใน "1" ใน C และจาก B ที่อยู่ใน "0" ใน C ในส่วน b แหล่งกำเนิด vectors A กำลังถูกอัดแน่นเพื่อให้ได้ผลลัพธ์ R ภายใต้ bit vector B การถูกอัดแน่นถูกทำใน B คำสั่งนี้มีประโยชน์ในต้นแบบของการเปลี่ยนแปลงสิ่งน้อยๆให้เหมาะสมอย่างยิ่ง

หลัก 3 อย่างของการเปลี่ยนหลักในสถาปัตยกรรมของ Star-100 ให้ผลกับ Cyber-205

1.ใน Star-100 ขั้นตอนของ scalar unit ถูกจับคู่เข้ากันด้วย vector unit ซึ่งเป็นเพียง 1 ประเภทของขั้นตอนจะสามารถแสดงออกมาได้ในเวลานั้น Cyber-205 มีควาสมสามารถในการ run ข้ออมูลทั้ง vector unit และscelar unit

2.ในแนวขนานกันรูปแบบการจัดการที่หลากหลายทางสิ่งแวดล้อม ก็คือตระกูล Cyber-200 จะมีขนาดหน้ากระดาษเล็ก เริ่มตั้งแต่ 4096 byte จนถึง 6553 bytes ส่วนหน้ากระดาษใหญ่ของ Star-100 ยังคงสภาพเดิมตั้งแต่มันมีเงื่อนไขสำหรับรองรับโปรแกรมขนาดใหญ่

3.ระบบ input/output ใน Star-100 เป็นชนิด "Star network" ด้วย node-to-node การสื่อสารระหว่าง CPU และส่วนรอบนอก การที่จะเปลี่ยน network จาก I/O หรือที่เรียกกันว่า loosely coupled network (LCN) เป็นสวิทซ์หลักสำหรับ hardware และ software ต่างจาก Cyber-205

4.1 Fujitsu VP-200 and Special Features

บริษัท Fujitsu ประกาศ FACOM vector processors VP-200 ในเดือน กรกฎาคม ปี 1982 ผลงานขั้นสูงในเครื่องกลนี้ บรรลุผลสำเร็จได้โดย เทคโนโลยี LSI, ปรับปรุงสถาปัตยกรรมให้ดีขึ้น, compiler ที่สลับซับซ้อน และจำนวนลักษณะที่ก้าวหน้าในทั้งพื้นที่ของ hardware และ software มันสามารถใช้เหมือน a loosely coupled back-end system. The block diagram ของ the VP-200 ถูกแสดงในรูป 4.33 ในระบบมี a scalar processor และ a vector processor ซึ่งสามารถปฏิบัติได้อย่างพร้อมกัน เหมือนใน the Cray-1 การลงทะเบียนมากและเร็ว buffers และ multiple pipes ถูกใช้เพื่อทำให้สามารถทำการ register-to-register operations. ความจำหลังมีตั้งแต่ 256 Mbytes ขึ้นไป ซึ่งเชื่อมต่อกับ the vector register via two load-store pipelines. แต่ละอย่างของ the two load-store pipes มีข้อมูล bandwidth ของ 267 M คำ ในทิศทางใดทิศทางหนึ่ง อัตรานี้จะเข้าคู่กับปริมาณในการผลิตที่มากที่สุดของ the arithmetic pipes. มี 4 การปฏิบัติงานของวิถีทางส่งข่าวใน vector processor รูปแบบข้อมูลสำหรับคำสั่ง vector สามารถเป็น bit strings, 32 bit fix-point,และ 32 หรือ 64 bit flating-point operands ได้มี 83 คำสั่ง vector และ 195 คำสั่ง scalar ใน the VP-200 ส่วนใหญ่ the scalar ในการจัดตั้งคือ IBM 370 ที่สอดคล้องกันและส่วนเชื่อมต่อกันของ the scalar unit ซึ่งความจำหลักผ่านทาง buffer storage ที่ใหญ่

เทคโนโลยี,สถาปัตยกรรม, compiling algorithms,และการดำเนินการของลักษณะของ hardware ที่คล้ายกันเหล่านี้ไปสู่ลักษณะเหล่านั้นใน the Cray-1 และ Cyber-205, และบางลักษณะถูกพัฒนาอย่างมีลักษณะเฉพาะตัวใน the Fujitsu machine

<u>Technologies utilized</u> เทคโนโลยีล่าสุดของ Fujitsu ถูกนำมาใช้เป็นประโยชน์ใน The FACOM vector processor. Logic LSIs บรรจุด้วย 400 gate/chip, บางหน้าที่พิเศษของ chip เช่น การปรรจุแฟ้มการลงทะเบียน 1300 gates. Propogation delay ที่สำคัญต่อ gate ของ LSIs 4 Kbit/module ด้วย an access time ของ 5.5 ns ถูกใช้ในที่มีความเร็วสูงอย่างยิ่งเป็นสำคัญ 121 LSIs สามารถตั้งอยู่บน a 14-layered printed circuit board เรียกว่า MCC (multichip carrier)

<u>Vectorizing complir-Fortran 77/vp</u> A vectorizing compiler,Fortran 77/VP, เคยพัฒนาสำหรับ the FACOM vector processor. Fortran 77 เคยถูกเลือกในฐานะที่เป็นภาษาสำหรับ machine นี้ ซึ่งประโยชน์ของ software ที่ใหญ่สามารถกลายเป็นสิ่งที่หาได้อย่างง่ายดาย ในการรับมาของอัตราส่วนของ high vectorization สำหรับแนวทางของโปรแกรมการประยุกต์ใช้, The Fortran

77/VP compiler vectorrize ไม่ใช่ the simple DO loops เท่านั้นแต่สร้าง DO loops และ the macro operation เช่น ผลผลิตภายในที่มีประสิทธิภาพ มันจะค้นพบและแบ่งแยกการเกิดขึ้นอีกซ้ำๆอีกด้วย.

Conditional vector operations การวิเคราะห์ของโปรแกรมการประยุกต์ใช้ ทำให้รู้ว่ารายการที่มีเงื่อนไขจะพบบ่อยภายใน DO loops.The FACOM vector processor ให้ 3 วิธีการที่แตกต่างกันเพื่อการปฏิบัติการของสาขาที่มีเงื่อนไขอย่างมีประสิทธิภาพ สำหรับ vector:masked arithmetic operations,compress-expand functions, และ vector indirect addressing.

ในการควบคุมการปฏิบัติการ vector ที่มีเงื่อนไขและหน้าที่การแก้ไข vector, bit strings (เรียกว่า mask vectors) ถูกให้มาด้วยผลรวมของ 256 mask registers,แต่ละ 32 bits, the store mask vectors และ the mask pipe แสดงการปฏิบัติการที่มีเหตุผล ซึ่งเกี่ยวข้องกับ the mask vectors.

Vector editing functions The FACOM vector processor ให้ 2 รูปแบบของหน้าที่การแก้ไข คือ compress-expand operations และ vector indirect addressing หน้าที่เหล่านี้ไม่สามารถใช้กับการปฏิบัติการของ vector ที่มีเงื่อนไขเพียงเท่านั้นแต่ใช้กับการคำนวณต้นแบบที่เล็กน้อยและการประยุกต์ใช้การแก้ไขข้อมูลอื่น ๆ Vectors บนการลงทะเบียน vector สามารถถูกแก้ไขโดยหน้าที่ย่อและขยาย โดยการใช้ load-store pipes เหมือนวงจรที่จัดข้อมูลให้อยู่ในแนวเดียวกัน ไม่มีทางเข้าสู่ความจำหลักซึ่งเป็นเหตุให้ยุ่งยากในกรณีนี้ การย่อ vector A หมายถึง ส่วนประกอบของ A ที่ถูกทำเครื่องหมายด้วย
1s ในที่ตั้งที่ตรงกันของ the mask vector ถูกคัดลอกเข้าไปใน Vector B อีกสิ่งหนึ่ง ที่ซึ่งส่วนประกอบเหล่านี้ถูกเก็บไว้ในที่ตั้งที่อยู่ติดกันด้วยคำสั่งของมันที่ถูกรักษาไว้ การขยาย a vector หมายถึงการปฏิบัติการในทางตรงกันข้ามกันเหมือนตัวอย่างที่แสดงข้างล่างนี้

Example 4.9

Vector registers optimization สิ่งหนึ่งของลักษณะส่วนใหญ่ที่ไม่ธรรมดาของ the FACOM vector processor คือ the dynamically configurable vector registers. แนวคิดของการลงทะเบียน vector มีความสำคัญมากสำหรับการแปรรูป vector ให้มีประสิทธิภาพ ตั้งแต่มันลดความถี่ของทางเข้าสู่ความจำหลักอย่างรุนแรง ผลลัพธ์ของการศึกษาของเราชี้ให้เห็นว่า ความต้องการสำหรับความยาวและจำนวนของ vectors แตกต่างจาก program หนึ่งไปสู่อีก program หนึ่ง การทำการใช้ให้เป็นประโยชน์ให้ดีที่สุดของขนาดรวมทั้งหมดของ 64 K bytes, การลงทะเบียน vector สามารถนำมาเชื่อมเข้าด้วยกัน ซึ่งนำมาจากรูปร่างภายนอกดังต่อไปนี้ คือ 32 (ความยาว) * 256 (จำนวน vector) 64*128, 128*64, …, 1024*8 ความยาวของการลงทะเบียน vector ถูกระบุโดยการลงทะเบียน hardware พิเศษ,และมันสามารถถูกเปลี่ยนแปลงได้โดยคำสั่งในโปรแกรม

High-level concurrency The FACOM vector processor อนุญาตให้เกิดขึ้นพร้อมกันที่ระดับต่างกัน ในหน่วย vector, 5 หน้าที่ของวิถีทางส่งข่าวสามารถจัดการได้โดยพร้อมกัน : 2 ทาง

ออกของ 3 หน่วยวิถีทางส่งข่าวที่เกี่ยวกับเลขคณิต, 2 load-store pipes,และ mask pipe ภายใจแต่ละหน่วยวิถีทางส่งข่าวที่เกี่ยวกับเลขคณิต, จำนวนของการคิดคำนวณ vector เกี่ยวข้องกับคำสั่งที่ต่อเนื่องกัน ซึ่งสามารถไหลได้อย่างต่อเนื่องโดยปราศจากการแตกกันของ the pipe.

หน่วย vector และหน่วย scalar สามารถลงมือทำงานได้โดยพร้อมกัน ซึ่งอธิบายในรูปภาพที่ 4.34 นอกจากลักษณะเช่นนี้แล้ว การปฏิบัติการ scalar ระหว่างการปฏิบัติการ vector จะเป็นสาเหตุที่ต้องคำนึงถึงการแสดงการลดขั้นคำสั่งการพิมพ์เป็นลำดับ ถูกให้เพื่อรักษาความสัมพันธ์ของข้อมูลไม่อิสระระหว่างคำสั่ง

5. VECTORIZATION AND OPTIMIZATION METHODS

ใน section นี้ เราจะศึกษา 4 ประเด็นสำคัญที่นำไปสู่การแสดงถึงการยกระดับของผู้แปรรูป vector เราเริ่มต้นด้วยการแนะนำของลักษณะภาษาที่เป็นประเภทเดียวกัน สำหรับการแปรรูป vector สิ่งที่สร้างขึ้นที่เป็นประเภทเดียวกันในการขยายออกของภาษาที่มีระดับสูงถูกอธิบายโดยตัวอย่าง แทนที่การอธิบายที่เป็นนามธรรม ขั้นตอนการออกแบบของ a vectorizing compiler สำหรับการผลิตรหัส vector จากแหล่งกำเนิดรหัสที่ถูกเขียนอย่างเกี่ยวเนื่องกัน เป็นการบ่งบอกลักษณะด้วยความง่ายดายต่าง ๆ ในการ vectorizing เมื่อเราศึกษาวิธีการ optimization ต่าง ๆ เพื่อผลิตรหัสที่มีประสิทธิภาพ ในที่สุด เครื่องมือที่เป็นการวิเคราะห์สำหรับการประเมินค่าการแสดงออกของวิถีทางส่งข่าวของ computer ที่ถูกนำเสนอ

5.1Parallel language for vector processing

ภาษาที่ผ่าน 2 กระบวนการ vector เคยเสนอเมื่อไม่นานมานี้ สิ่งหนึ่งคือ the Actus โดย Perrott(1979) และอีกสิ่งหนึ่งคือ the Vectran โดย Paul และ Wilson(1975) อย่างโชคร้าย, ไม่ใช่ทั้ง 2 ของภาษาที่เป็นประเภทเดียวกันที่ประสบความสำเร็จในการทดลองกับ real machine ภาษาที่เป็นประเภทเดียวกันจะไกลจากการทำให้ได้มาตรฐาน ลักษณะที่ปรารถนาประกอบด้วยลักษณะที่เปลี่ยนแปลงได้ ในการอธิบายและการเลือกจัดกระบวนการของวัตถุใน columns,rows,blocks,diagonals,และในการแสดงออกของการจัดกระบวนการย่อยต่างๆ ; ความมีประสิทธิภาพในการเปลี่ยนแปลงน้อย ๆ และ ต้นแบบที่หนาแน่น : การปฏิบัติตามการจัดกระบวนการ เพื่ออนุญาตให้พอที่จะเคลื่อนย้ายได้ และวิธีการดำเนินงานเพื่อหยุด vectorization barriers

ความมีประโยชน์ของภาษาใหม่ขึ้นอยู่กับพื้นที่ที่ประยุกต์ใช้ของมัน เราจะย่อให้เป็นลักษณะที่อยู่เบื้องล่างโดยการขยายออกของ Fortran ตัวอย่างเช่น บางลักษณะที่ดึงดูดความสนใจในแบบฉบับของภาษาที่เป็นประเภทเดียวกัน A vector บางทีจะแสดงตัวอย่างแน่นอน โดยลักษณะท่าทางของชื่อที่จัดกระบวนการถูกตามโดยสัญลักษณ์หรืออักษรที่เขียนอยู่ในระดับต่ำกว่าอักษรปกติ การใช้เครื่องหมายขยายออก บางทีจะชี้เฉพาะเจาะจง an implied DO notation อย่างถ้วนทั่ว เช่น

$e_1 : e_2 : e_3$

$e_1 : e_2$

*

$e_1 :$ * $: e_3$

เทคนิคอื่น ๆ เช่น การจัดสรรการลงทะเบียน, vector hazard, และคำสั่งการจัดเรียงใหม่ คือ machine ที่ไม่อิสระ ตัวอย่างเช่น, เราต้องการที่จะจัดสรรการลงทะเบียน vector ใน the Cray-1 เพื่อให้มีผลในเวลาของการปฏิบัติงานที่น้อยที่สุด การจัดเรียงการปฏิบัติงานลำดับเหตุการณ์ เพื่อปฏิบัติขั้นตอน vector ที่เหมือนกันอย่างช้า ๆ สามารถลด the pipeline reconfiguration overhead ใน a multifunctional pipe A vectorizer ให้รายละเอียดกับ programmer ของความเป็นไปได้ของขั้นตอนที่เป็นประเภทเดียวกัน มันจะให้เครื่องมือในการเรียนรู้ด้วย ซึ่ง programmer สามารถตรวจสอบข้อมูลที่ส่งออกมาของ the vectorizer และปรับแต่งการคำนวณสำหรับวิถีทางส่งข่าวที่ดีกว่า Automatic vectorization และ code optimization จะเพิ่มความสามารถในการผลิตของ vector processors.

5.2 Design of a Vectorizing Compiler

vectorizing compiler สามารถวิเคราะห์ stetement ใน loop Do ได้ และในทำนองเดียวกันก็สามารถ execute object code และ vector insturction ได้เช่นกัน อัตราส่วนของ vectorization มีความสำคัญกับประสิทธิภาพ เพื่อผลสำเร็จนี้ compiler vectorizes จึง access ข้อมูลและหาผลลัพธ์ของโปรแกรมได้ยุ่งยากและซับซ้อน ซึ่งขึ้นอยู่กับการฝืนความควบคุมของ machine hardware

อุปสรรคของ vectorization อยู่ภายใต้เงื่อนไขและแบ่งออกเป็น statement เทคนิคการออกแบบ vecterizing compiler สามารถอธิบายได้คร่าวๆดังนี้

1. ถ้ามองกลับไปยังขั้นตอนปกติของFortran compiler อย่างหนึ่งคือ ขั้นตอนของ lexical-scan และsyntax-passing ที่เปลี่ยน source program เป็น intermedate code,quadruples เป็นต้น ในการใช้มีการกำหนด field เป็น 4 เท่า 2 operand , 1 operand สำหรับผลลัพธ์เสมอ และaxiliary field จะช่วย 2 ขั้นตอน คือ code optimization และ code ganeration จุดประสงค์ที่กำหนด operand คือเพื่อหาใน register ว่าอยู่ใน register ไหน , คำสั่งไหนควร delete } และอื่นๆ

ตัวอย่าง A= A+B*C

$T1 \leftarrow B*C$

T2 ← A+T1

T1 คือการระบุเป็นตัวแปรชั่วคราว

2. compiler จะยอมรับ input 4 เท่า และการเปลี่ยนแปลงผลของ output 4 เท่า เช่นกัน optimizer จะเป็นผู้หา subexpression B*C ก่อนแล้วนำมาคำนวณและเรียก B*C ว่า T2

T1 ← T2

A ← A+T1

Optimizerจะไม่นำ T1 มาใช้อีก ดังนั้นมันจะถูกแทนที่ด้วย T2 ทันที และผลลัพธ์ก็จะมีเพียง statement เดียว คือ A ← A+T2 register จะ allocate ไปที่ CPU ถ้ามันพบ A ที่ใช้ มันก็จะassign ให้ rregister นั้น คือ R3 และจะบันทึกข้อมูลลงไป

A(R3) ← A(R3) +T2

3. โดยทั่วไปแล้วจะ Tranfer ข้อมูลสุดท้ายคือ intermedia code เป็น machine-language พวกเราสามารถเขียนในรูปของ machine code คือ ADD R3,M(T2)

M(T2) หมายถึงพื้นที่หน่วยความจำของ T2

ใน vectorizing Fortran compiler,scanner และ parser ต้องการเปลี่ยนแปลง scalar code แล้ว พวก

เราก็ต้องเปลี่ยน scalar operation เป็น vector code

เรามาพิจารณาเทคนิคบางอย่างของ optimization ว่าเทคนิคไหนที่เราสามารถขยายหรือเพิ่ม intermedia code ได้ ตั้งแต่ที่เราต้องพึ่งเทคนิคของ optimization เราจะพิจารณาโครงสร้างของ Cray-like ในทั้งหมดของ vector operation คือ register-to-register

5.3 Optimization of Vector Function

เราจะอธิบาย 9 ตัวอย่างสั้นๆของ vector function บางอย่างก็เป็นเทคนิคที่ต้อง apply ดังนี้

(a) **Redunddant expression elimination** หลังจากที่ทำการหาเครื่องหมายที่เกินความต้องการใน intermedate code แล้ว เราควรกำจัดมันทิ้ง ซึ่งทำให้ตัวเลขของ memory access และ execution tiom ลดลงด้วย

(b) **Constant folding at compile time** constant folding หมายถึง การคำนวณจาก runtime ไป compile time แม้ว่าโอกาสทำoperation บน array ที่คงที่ก็ไม่บ่อยครั้งนัก เช่น ใน loop Do ของ array ทั่วๆไป

A(I)=I for I=1,2,3,..,100 สามารถหลีกเลี่ยงขั้นตอนการ execute เพราะ array A(I) เป็น vector คงทีjj(1,2,3,..,100) ได้ ใน compile time

(c) **Invariant expression movement** ใน loop ของ vector operation ในบางกรณีสามารถ replaced บางvector ถ้าไม่มี data ถ้าบาง statement คือ loop-variant , loop-invariant expression จะออกจาก loop แบบนี้เราเรียกว่า " code motion " ตัวอย่าง loop two-level

```
DO 20 I = 1,n
    .....
DO 20 J=1,m
    .....
B(I,J)=B(I,J) +A(J)*C(J)
20 CONTINUE
```

(d) **Pipeline chianing and parallelization** ใน vector processor กับ multiple functional pipe , ประสิทธิภาพสามารถ update จาก chianing several pipelines ผลลัพธ์ จาก 1 pipeline จะ direct input ให้ pipeline อื่นๆ เวลาที่ให้ stroring ผลลัพธ์ คือ thus eliminated ผลลัพธ์ไม่ต้องการ Storedd กลับไปใน memory กับ chianing compiler ที่ดี ควรจะมีความสามารถที่จะได้ผลของ operation ที่สามารถ chained อย่างถูกต้องเารกล่าวได้ว่า บาง chaining operation ของการคูณ pipe ในหัวข้อ 4.1 แล้ว

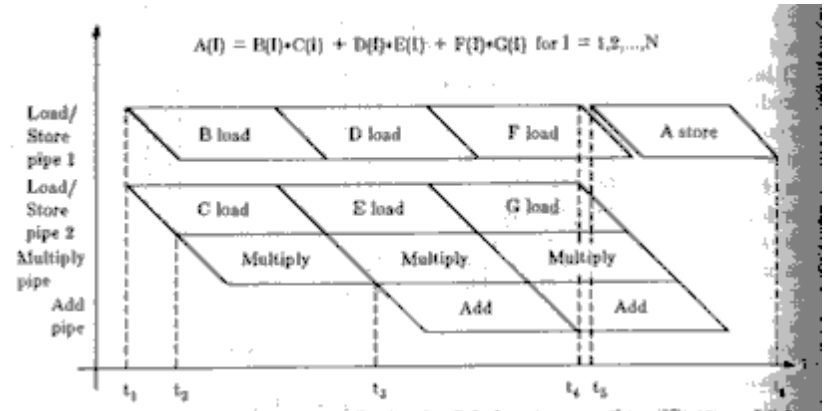


Figure 4.37 Pipeline chaining and parallelization for linked vector operations. (Courtesy of Fujitsu Limited, Japan.)

(e) **Vector register allocation** บน machineจะยอมรับ Cray-1 ของ vector operation ความสำคัญของ register allocation ไม่สามารถเกินได้ การได้รับการคำนวณมากที่สุด execution unit จะทำ operand ต่อไปเรื่อยๆ vector ใน register ระหว่าง operation คือ 1 ทาง ที่เราจะต้องเจอ อย่างไร

ก็ตามวิธีของ vector register จะเน้นการแบ่ง local และ global ออก เพราะ ขอบเขตของระบบตัวเลขนั้นๆแตกต่างกัน

ตัวอย่าง A-B*C สามารถ executed ใน Cray-1 จากการใช้ vector register 3 ตัว คือ V1,V2,V3

V1 ← A

V2 ← B

V3 ← C

V2 ← V2*V3

V1 ← V1-V2

ถ้าเราคูณก่อน looding vector A ไป register จะมีแค่ 2 vector register เท่านั้น

V1 ← B

V2 ← C

V1 ← V1*V2

V2 ← A

V2 ← V2-V1

(f)Reorder the execution sequence

A ← A2+A3		A1 ← A2+A3
A4 ← A5*A6		B1 ← A7+A8
B1 ← A7+A8	--→	B5 ← C1+C2
B2 ← B3*B4		A4 ← A5*A6
B5 ← C1+C2		B2 ← B3*B4

ผลอยู่ทางซ้าย ต้องการโครงสร้าง pipeline 3 ตัว ทางขวา ต้องการแค่ 1ตัวเท่านั้น compilerที่ดีควรจะ reorder execute sequence ให้ได้ตัวเลขของการ require pipeline ให้น้อยที่สุด

(g)Temporary storage management ในขั้นตอนของ optimization ของ vectorizing compiler โดยทั่วไปแล้วintermedate vector ปริมาณมาก สามารถแก้ปัญหาสำคัญๆหรือหนักๆได้

ตัวอย่างของการ execute คำสั่งของ vector ใน cray-1

A(1:4000)=A(1:4000)*B(1:4000)+C(1:4000)

ต้องการ vector register 63 กับ 64 component ซึ่งนำมายึดชั่วคราว กับผลิตภัณฑ์ของ intermedate ทั้งหมด

ผลลัพธ์ของ intermedate จะเก็บไว้ชั่วคราวที่ memory และเหตุผลนี้ compiler จะ allocate และ deallocate พื้นที่ไว้ชั่วคราว

(h)Code avoidance มาถึงพื้นฐาน optimization ของ vector operation คือ เทคนิการ copy optimizatiion ความคิดนี้คือการหลีกเลี่ยงการ copy arry ที่มากเกินไป ที่กำลังไม่น้อยกว่า arryที่จะ copy มากเกินไป จึงจะทำ

พิจารณา code sequence

A(1:50)=B(1:99:2)

….

….

C(1:50)=2.0*A(1:50)

ถ้า compiler adjusts storage-mapping function สำหรับ array A หมายถึง storage สำหรับ array B

V1 ← B(1:99:2)

A(1:50) ← V1

…

…

V1 ← A(1:50)

V1 ← 2.0*V1

C(1:50) ← V1;

จะได้ V1 ← B(1:99:2)

V1 ← 2.0*V1

C(1:50) ← V1;

(i) Turning for interaction การส่งคืนเครื่องมือ คือความจำเป็นที่จัดเตรียมเพื่อ user interaction ใน vectorization Cray-1 และ VP-200 มีเครื่องช่วยในการส่งคืนใน VP-200 จาก display เครื่องมือที่จะส่งข้อมูล และ vectorizing effect คืน user สามารถ modify source program จะ optimized ไปทาง vectorization ที่เต็ม จาก vectorizing compiler number ของcompiler directive line ยังใช้ check ได้ ไม่ว่าจะใน source code, true ratioใน IF statement ,vector length distribution และอื่นๆ

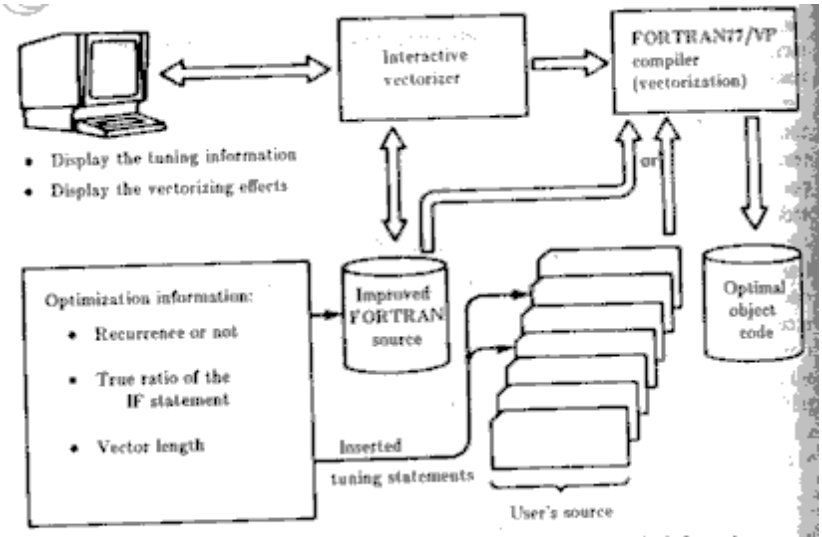


Figure 4.38 The tuning facilities in FACOM VP-200. (Courtesy of Fujitsu Limited, Japan.)

optimal object code with vector instructions after the tuning process is completed. An example is given below to illustrate the tuning concept realized in the VP-200

5.4 Performance Evaluation of Pipeline Computer

ใน section นี้ เราจะพูดถึงประสิทธิภาพของ pipeline vector processor ประสิทธิภาพจะขึ้นอยู่กับ processor utilization และ speedup ประสิทธิภาพของ pipeline ขึ้นอยู่กับ pipeline rate ,การ Lood ข้อมูล และ utiliZation rate ของระบบ resources

การคำนวณหาประสิทธิภาพ

K: จำนวนของ storage ใน function pipe

T: ผลรวมของ pipeline delay ใน คำสั่ง ที่ execute

Ni:ความยาวของ ตัวแปร vector ที่ใช้ในคำสั่ง (1<=I<=n)

N: จำนวนของคำสั่งที่ contained ใน program

W: throughput ของ pipeline computer

Ti: เวลาการ required จนทำคำสั่งเสร็จใน pipelinecomputer (1<=I<=n)

Tp:ผลรวมของเวลาในการ required จนเสร็จการ task consisting ของคำสั่ง n

Sk: speedup ของ pipeline computer

η: ประสิทธิภาพของ pipeline computer

τ: T/K

$$Ti = (K-1)\tau + Ni(\tau) = (Ni+K-1)(T/K)$$

6. SIMD ARRAY PROCESSOR

ชุดสัญญาณนับได้จังหวะของการประมวลผลแบบขนานเรียกว่า array processor ซึ่งเป็นการรวมของการประมวลผลหลายๆตัวของ PE ภายใต้การควบคุมของ CU (Control Unit) Array Processor สามารถทำให้เกิด single instruction และ mutiple data streams (SIMD) ซึ่งถูกสร้างขึ้นมาเพื่อทำการคำนวณ Vector บนMatrices หรือชุดของข้อมูล

SIMD ปรากฏใน 2 พื้นฐานของระบบสถาปัตย์ Array processor ใช้หน่วยความจำ rondom-access และ associative processors ใช้หน่วยความจำ Content-addressable (หรือ associative) แบ่งออกเป็น 3 ส่วน สำหรับ primary และ array processor

Later-PE Communication

การออกแบบ network เป็นตัวตัดสินใจสำหรับการติดต่อแบบ inter-PE สิ่งเหล่านี้เป็นราก,ฐานสำคัญสำหรับการตัดสินใจในการออกแบบได้อย่างเหมาะสม สถาปัตยกรรมของการเชื่อมต่อระบบเครือข่ายสำหรับโครงสร้างของ SIMD แบ่งออกเป็น 2 กลุ่มคือ

1.Operation models, control strategies, switching methodologies

2. Network topologies

Operation mode มี 2 ชนิดที่ใช้ในการติดต่อที่สามารถระบุได้คือ synchronous และ asynchronous

synchronous เป็นการติดต่อที่จำเป็นสำหรับเส้นทางในการติดต่อสื่อสารสำหรับของแต่ละฟังก์ชั่น ในการส่งผ่านข้อมูลทั่วไป

asynchronous เป็นการติดต่อที่จำเป็นสำหรับ multi-processing ที่ซึ่งเป็นกราติดต่อทีต้องการแบบ issued dynamically ระบบ Operation mode สามารถที่จะสร้างอย่างง่ายๆทั้ง asynchronous และsynchronous processing

เพราะฉะนั้นชนิด Operation mode ของการเชื่อมต่อเครือข่ายสามารถแบ่งได้ 3 ประเภทคือ

1. synchronous
2.asynchronous
3. Combined

โครงสร้างของ SIMD ที่มีทั้งหมดเลือกใช้ synchronous operation mode ซึ่งใน synchronous operation จะมี lock-step operation ซึ่งจะต้องใช้ใน Pes

Control strategy เป็นชนิดการเชื่อมโยงการติดต่อระบบเครือข่ายประกอบด้วยตัวเลของ Switching elements และ Interconnecting links ฟังก์ชันของการเชื่อมโยงติดต่อโดยเกี่ยวกับการติดตั้งระบบควบคุมของ Switching elements การติดตั้งการควบคุมฟังก์ชันสามารถจัดการโดยผู้ควบคุมส่วนการกลางหรือโดยเฉพาะบุคคล Switching elements latter strategy ถูกเรียกว่า Distributed control แบะแผนการอันดับแรกที่เหมือนกันของ Centralized control ส่วนมาก SIMD ที่มีอยู่จะ

เชื่อมโยงระบบเครือข่ายที่ใช้เลือกการควบคุมระบบส่วนกลางบน Switching elements ทั้งหมดโดย CU(control unit)

Switching methodology มี 2 ตัวคือ circuit switching และ packet switching

-*circuit switching* คือเส้นทางหลักระหว่างแหล่งที่มาและจุดมุ่งหมาย

-*packet* switching คือข้อมูลที่นำมาไว้ในหีบ และเส้นทางผ่านการเชื่อมโยง network นอกจากเส้นทางการเชื่อมโยงหลักของ physical ปกติ circuit switching ส่วนมากเหมาะสมสำหรับการส่งผ่านข้อมูลขนาดใหญ่ และ packet switching มีประสิทธิภาพสำหรับข้อมูลสั้นๆ

ทางเลือกอื่น เช่น integrated switching เป็นความสามารถในการรวมกันของ circuit switching และ packet switching ด้วยเหตุนี้สามารถระบุ Switching methodology ได้ 3 ตัวคือ circuit switching , packet switching, integrated switching

SIMD ส่วนมากเชื่อมโยงเครือข่าย network จะถูกเชื่อมต่อโดย handwired ไปยังการทำงานของ circuit switching ขนาดใหญ่ ส่วน packet switching ถูกมองรวมเป็นส่วนหนึ่งในโครงสร้างของ MIMD

<u>Network Topology</u> ระบบ network สามารถอธิบายวาดให้เห็นกราฟ ที่ซึ่งเป็นตัวแทนของ switching points และเป็นเส้นที่แสดงสายการเชื่อมโยงกัน

Topologies สามารถรวมเป็น กลุ่ม ได้ 2 ประเภทคือ static and dynamic

-*static topology* เส้นที่เชื่อมระหว่าง processorสอง ตัวคือ จะคงที่และ dedicated bus ไม่สามารถ reconfigured สำหรับการติดต่อโดยตรงไปยัง processor ตัวอื่น บนหัวของ processor ตัวอื่น เส้นใน dynamic cetology สามารถ reconfiguredโดยการติดตั้ง การทำงานของระบบเครือข่าย switching elements ที่ว่างของเครือข่างการติดต่อสามารถใช้เป็นตัวแทนโดย cartesia poduct ของลักษณะการออกแบบ 4 เซตข้างล่าง

(operation mode) x (control strategy) x (switching methodology) x (network topology)

ไม่ทั้งหมดของการรวมกันของลักษณะการออกแบบที่น่าสนใจ ทางเลือกของเฉพาะ interconnection ขึ้นอยู่กับ การประยุกต์คำสั่ง,เทคโนโลยีที่ใช้ และราคาที่ทำให้ได้ผลประโยชน์

<u>6.1SIMD INTERCONNECTION NETWORKS</u>

Interconection network ต่างๆจะถูกมองรวม สำหรับ SIMD ในส่วนนี้เราจะแบ่งออกระหว่าง single-stage . recirculating network multistage SIMD network ชั้น network ที่สำคัญประกอบด้วย

1. The 1lliac
2. The flip network
3. The n cube
4. The Omega
5. The data manipulator

6. The barrel shifler
7. Shuffle-exchange network

6.2 Connection Issues for SIMD Processing

ชุดการทำงานของ SIMD จะแสดงออกมาอย่างชัดเจนคล้ายกับโปรแกรมผู้ใช้คอมไพล์จะตรวจหาความคล้ายคลึงกันและทำให้เกิดโค้ดสำหรับที่จะทำ(excution) ในการประมวลผลหลายโปรแกรมและ Control Unit ส่วนของโปรแกรมที่ซึ่งไม่สามารถครอบคลุมไปในรูปแบบในการทำงานแบบขนาน คือจะส่งไปใน PE และทำ synchronously บนข้อมูลที่ถูกดึงมาจากหน่วยความจำขนานใต้ตัวควบคุมของ Control Unit สัญญาณ synchronous สามารถถ่ายเทไป PE ข้อมูลจะ permuted และจัดเรียงในรูปของ vector เช่น เมื่อเรารันโปรแกรมผลที่ได้ส่วนมากจะอยู่บน array processor หนึ่งจะต้องพัฒนาเทคนิคสำหรับ vectorizing โค้ดโปรแกรม เครือข่ายการติดต่อจะให้บทบาทหลักใน vectorization การเชื่อมต่อต่างใช้ SIMD ในการเชื่อมต่อระบบเครือข่าย คือ address ข้างล่าง

Dermutation and Connectivity ใน array processing ข้อมูลมักจะถูกเก็บไว้ในหน่วยความจำขนานในรูปแบบที่บิดเบือน นั้นคือยอมให้ vector ของข้อมูลไปดึงส่วนตรงข้าม อย่างไรก็ดี การดึงข้อมูลต้องเป็นข้อมูลจริงในการกำหนดแบบแผนก่อนที่มันจะสามารถส่งไปแต่ละ PE สำหรับการประมวลผลนี้เป็นวิธีการจัดโดยเส้นทางฟังก์ชันของการเชื่อมต่อเครือข่ายที่ซึ่งจะทำให้เกิดข้อมูลโดยแต่ละ PE ไปในรูปแบบที่ผิดออกไปสำหรับการเก็บข้อมูลในรูปแบบทีผิดออกไปสำหรับการเก็บข้อมูลในรูปปบบหน่วยความจำ memory modules

การปรับปรุง network และ nonblocking network สามารถให้ฟังก์ชัน ในการแทนที่ได้ แต่การใช้ network สำหรับการจัดเรียงต้องการตัดสินใจของตัวควบคุมการคำนวณ วิธีการเรียกตัวเองซึ่งสามารถมองรวมสำหรับ 2-3 ครอบครัวของการเปลี่ยนที่จำเป็นสำหรับการประมวลผลแบบขนาน อย่างไรก็ตามปัญญาก็ยังมีอยู่สำหรับความเข้าใจของการเปลี่ยนที่โดยทั่วไป พยายามที่จะทำบนความสามารถในการเปลี่ยนที่ของ single-stage network และblocking-multistage network ไม่สามารถที่จะเคลื่อนที่ได้ตามใจใน single pass เป็นผลลัพย์ที่แสดง baseline network สามารถเคลื่อนที่ได้ตามใจใน two passes ในขณะที่ blocking อื่นๆมี networks หลายขั้นตอนซึ่ง Omega network จำเป็นอย่างน้อย 3 เส้นทาง

Partitioning and reconfigurabilty

รูปแบบโครงร่างที่ถูกเสนออย่างดีใช้การเชื่อมโยง network ภายใต้ concept นี้ networkจะพอเหมาะกับโครงร่างในเชิงเรขาคณิตที่เหมือนกัน โครงร่างที่ใช้ในฟังก์ชันที่ใช้ในการเปลี่ยนที่ที่ติดต่อกับการเชื่อมโยง network เราสามารถที่จะกำหนดชื่อ input-output link ในฟังก์ชันการเปลี่ยนที่ในหนึ่ง conflict-free pass กำหนดชื่อ logical ของฟังก์ชันการเปลี่ยนที่ที่แตกต่าง เรียกว่า ปัญหาในการเรียก

โครงร่างเดิม ผ่านการประมวลผลเดิม baseline network สามารถที่จะเปลี่ยนที่ได้ทุกๆครั้งใน one pass ความหมายของการประมวลผลเกิดขึ้นพร้อมกันตั้งแต่ต้นจนจบสามารถที่จะเปลี่ยนแปลงโดยการกำหนดงานที่เหมาะสมด้วยการประมวลผลจากตัวเลขและข้อมูลในหน่วยความจำ

เมื่อแบ่ง SIMD ที่เชื่อมโยงกับระบบเครือข่ายที่มีขนาด subnetwork ที่แตกต่างกัน subnetwork จะสามารถเชื่อมโยงกับระบบเครือข่ายของ network ได้อย่างสมบรูณ์แบบที่มีชนิดและขนาดเหมือนกัน Hence กับการแบ่งข่ายของ network ระบบนี้สามารถช่วยเหลือโครงสร้างหลายๆตัวของ SIMD โดยการเคลื่อนที่แบบโครงร่างเดิมของ dynamical ซึ่งมีระบบโครงสร้างอิสระของ SIMD และการแบ่งงานที่เหมาะสม การแบ่งเป็นส่วนๆนี้ เราสามารถหาแหล่งที่มาอย่างมีประสิทธิภาพได้ Singel(1980) สามารถปรับปรุงระบบเครือข่ายของ single-stage ที่เปลี่ยนแปลงได้และ Illac networks ไม่สามารถที่จะแบ่งตัวแทนของระบบ network ได้อย่างอิสระ แต่ blocking หลายขึ้นในชนิด network ซึ่งผู้จัดการข้อมูลสามารถแบ่งเป็นส่วนๆก็ได้

Reliability and bandwidth ความไว้วางใจการทำงานของระบบเชื่อมโยง network เป็นสิ่งสำคัญของระบบประสิทธิภาพทั้งหมด ความไว้วางใจในการออกคำสั่งมีอยู่ 2 ปัญหาคือ การวินิจฉัยที่ฟิดและ fault toterance

-การวินิจฉัยที่คิดเป็นปัญหาที่เกิดขึ้นบนระบบเชื่อมโยง network หลายๆขั้นตอนของ switching elements พบอยู่ 2 ขั้นตอน ปัญหานี้เกิดขึ้นจาก fault-detection และ fault tolerance เป็นการออกแบบ network ที่สำคัญซึ่งเป็นความสามารถในการเชื่อมโยงทั้งหมดมารวมกันกับการลดขั้นสิ่งที่ผิดที่มีอยู่

- network ที่สูงจะมีแถบความถี่กว้างซึ่งย่อยครั้งที่มีความต้องการ stone work ราคาต่ำ ราคาของ network ส่วนใหญ่ถูกกำหนดโดยความซับซ้อนของ switch รวมถึงการออกแบบ Cost-effective network ซึ่งในการค้าพบมากกว่า 7 อยู่ระหว่างประสิทธิภาพ (bandwidth) และความซับซ้อน (complexity) ซึ่งอุปสรรคหลักของ bandwidth ซึ่งขึ้นอยู่กับการวิเคราะห์ที่ไม่สามารถคาดเดาได้ของโปรแกรม และทำให้ง่ายมากมายในโปรแกรมประยุกต์ และ reliability ในข้อบกพร่องในการบรรลุผลงานสำหรับทั้ง SIMD และ MIMD Computers

<u>The spece fo SIMD computer</u>

ในส่วนนี้เราจะพิจารณาหลักของ SIMD นั้นคือได้สร้างออกแบบหรือเสนอขึ้นจนถึงปี 1983 เราใช้ในรูปของ array processor ซึ่งไม่รวมสำหรับ SIMD ที่ใช้ตามแบบ random-access memory และในเทอมของ associative processor สำหรับ SIMD ใช้หน่วยความจำ associative

เราแบ่งที่ว่างของ SIMD ไปใน 5 ที่ว่างย่อย,ฐานบน word-slice และการประมวลผล bit-slice และตัวเลขของ Control Unit ใช้

Word-slice array processor

Bit-slice array processor

Word-slice associative processors

Bit-slice associative processors

Multiple-SIMD computer

6.3 Array and Associative Processors

ในปี 1958 Unger ได้คิดโครงสร้างของคอมพิวเตอร์สำหรับปัญหาเกี่ยวกับระยะ array 2 มิติของ PEs ถูกควบคุมโดยคำสั่ง master เครื่อง Unger จะรองรับสำหรับแบบ recognition application (การประยุกต์การจำ)

คอนเซปท์ของ lock-step SIMD operation คือ ทำให้แข็งแรงใน solomon คอมพิวเตอร์ได้ถูกรองรับโดย Slotnick

-ในปี 1962 เครื่องคอมพิวเตอร์ Solomon แม้ว่าไม่เคยสร้าง motivated การพัฒนาของ Illiac series และเครื่อง later SIMD ก็ยังคงมีอยู่ แสดงใน figure 6.1

-ในปี1965 Senzig และ Smith ได้ออกแบบ Vector arithemtic กับมาตราฐาน memory modules และมาตราฐานสถาปัตย์ของ pipeline อย่างเช่นอธิบายใน figure 6.2 แต่ละ PE ใน virtual processor มีเพียง 2-3 รีจิสเตอร์ที่ทำงานบน PE ชุดการประมวลผล pipeline ถูกสร้างเพื่อเก็บฮาร์ดแวร์ใน vector processing

The Illiac-IV ประดิษฐ์มาจากผู้ประดิษฐ์หลายๆคนในการประดิษฐ์ Illiac series มันเป็นรุ่นแรกของชุดซุปเปอร์คอมพิวเตอร์ซึ่งพัฒนาต่อมาในปี 1960

ในปี 1979 ข้อเสนอ 2 ข้อได้ขยายสถาปัตย์ Illiac IV-BSP เพื่อตอบสนองความต้องการในอนาคตของเครื่อง gigaflop โปรเจค phoneix ได้รวมเอา SIMD หลายๆตัวประกอบเป็นชุดของ 16 Illiac-IV รวมกับ 1024 PEs Burroughs รองรับสถาปัตย์กับการอัพเกรด BSP ไปยัง 512 Pes แบ่ง 521 memory modules Burroughs

การประมวลผลแบบต่างๆของ bit-slice พัฒนาในยุโรปและอเมริกา ล่าสุด CLIP-4 เป็นการประมวลผลที่สร้างแบบ 96 X 96 PEs กับ 8 ตัวอื่นต่อ PEs CIIP ถูกออกแบบสำหรับ bit-slice ที่ใช้รูปแบบการประมวลประยุกต์ The Distributed Array Processor (DAP) ได้ถูกพัฒนาโดย International Computer limited ในประเทศอังกฤษ The DAP เป็นโครงสร้างในกลุ่มของ 16 Pes ในขนาดต่างๆ อย่างเช่น 32 X 32, 64 X 64,
128 X 128 และ 256 X 256

-The MPP เป็นตัวแทนของชุดโครงสร้าง State-of-the-art ผิวขนาดใหญ่ของ SIMD ในปี 1980 มันจะถูกนำไปใช้เกี่ยวกับดาวเทียมของ NASA

6.4 SIMD Computer Perspective

มันคือ SIMD ที่พิเศษรองรับระบบ สำหรับปัญหาแวดล้อม พวกมันอาจจะทำให้มีปัญหาชุดประมวลผลมีบางโปรแกรมและปัญหา vectorization ที่ซึ่งยากจะแก้ปัญหาจริงๆแล้ว การประมวลผลแบบนี้ไม่เป็นที่นิยมในหมู่ของเครื่องคอมที่ใช้ในการค้า

Performance ของชุดาการประมวลผล จุดที่ดีที่สุดคือ การแสดงภายใต้แนวคิดของโปรแกรมและแหล่งที่ทำให้เกิดเงื่อนไข เช่น ขนาดของชุด PE ที่เพิ่มขึ้นการทำก็ควรจะเพิ่มขึ้นด้วย แน่นอนว่าจุดที่ดีมักจะมีฟังก์ชันของระยะโดยเฉพาะสำหรับการทำงาน bit-slice สำหรับการประมวลผล vector มีประสิทธิภาพมักจะขึ้นอยู่กับความยาวของ vector

Multiple-SIMD (MSIMD) รูปแบบของชั้นย่อยของ MIMD โครงสร้างต่างๆ เป็นส่วนที่มีอยู่ใน multiple-array processor แต่ละโครงสร้างจะมีชุดของข้อมูลอย่างเช่นเดียวกับใน SIMD The Illiac-IV เป็นต้นแบบที่ใช้กับเครื่อง MSIMD สิ่งเหล่านี้มักจะมีใน MSIMD ตัวอื่น เช่น โปรเจค phonix และ the PM MSIMD สามารถอำนวยให้การประยุกต์ได้ยืดหยุ่นสูงหลังจากสามารถใช้ SIMD อย่างเดียว

ชุดข้างล่างนี้บางอย่างการประยุกต์พื้นที่หรืออาจรวมสำหรับชุดการประมวลและโดยเฉพาะสำหรับ Illiac-IV , The BSP, The MPP และระบบ STARAN

Matrix algebra (Multiplication, decomposition and inversion)

Matrix eigenvalue calculations

Linear and integer programming

General circulation weather modeling

Beam forming and convolution

Filter and fourier analysis

Image processing and pattern recognition

Wind-tuned experiments

Automated map generation

Real-time scene analysis

บางอย่างของการประยุกต์เหล่านี้อาจจะเคยเรียนกับการอธิบายเกี่ยวกับสถาปัตย์คอมในส่วนที่ผ่านมา ที่กล่าวมาข้างต้นไม่ได้หมายถึงทั้งหมดที่มี ส่วนมากการประยุกต์เหล่าจำเป็นต้องนำไปใช้ในส่วนของข้อมูล