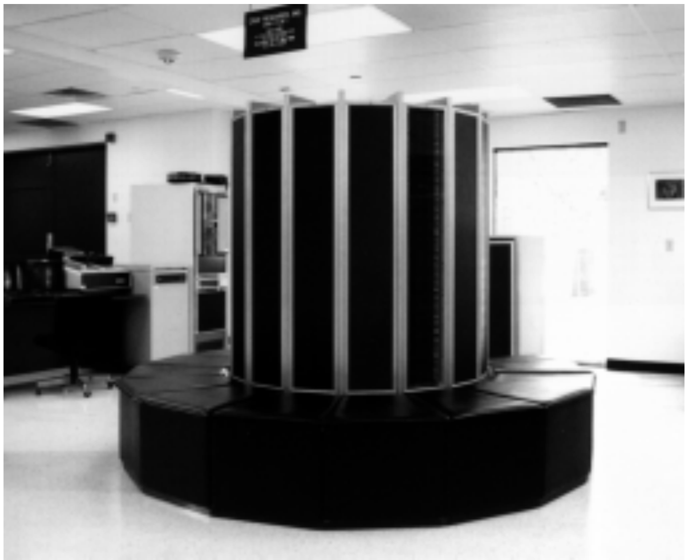
Cray-1 History

- In January 1978, a CACM article says there are only 12 non-Cray-1 vector processors worldwide:
  - Illiac IV is the most powerful processor
  - TI ASC (7 installations) is the most populous
  - CDC STAR 100 (4 installations) is the most publicized
- Recent report says the Cray-1 is more powerful than any of its competitors
  - 138 MFLOPS for sustained periods
  - 250 MFLOPS for short bursts
- Features: chaining (access intermediate results w/o memory references), small size (allows 12.5 ns clock = 80 MHz), memory with 1M 64-bit words

Cray-1 Physical Architecture

- Physical architecture
  - “World’s most expensive love-seat”
  - Cylindrical, 8.5’ in diameter (seat), 4.5’ in diameter (tower), 6.5’ tall (tower)
  - Composed of 12 wedge-like columns in 270° arc, so a “reasonably trim individual” can get inside to work
  - “Love seat” hides power supplies and plumbing for Freon cooling system
- Freon cooling system
  - In each chassis are vertical cooling bars lining each wall
  - Freon is pumped through a stainless steel tube inside an aluminum casing
  - Modules have a copper heat transfer plate that attaches to the cooling bars
  - 70F tube temp = 130F center of module

Cray-1



Cray-1 Architecture

- Computer architecture
  - 12 I/O channels, 16 memory banks, 12 functional units, 4KB of register storage
  - Only 4 chip types
  - Fast main memory, fast computation
- 4 chip types
  - 16x4 bit register chips (6 ns)
  - 1024x1 bit memory chips (50 ns)
  - Simple low- or high-speed gates with both a 5-wide and a 4-wide gate (5/4 NAND)
- Fabrication
  - 6”x8” printed circuit boards
  - ICs in 16-pin packages, up to 288 packages per board to build 113 different module types, up to 72 modules per 28-inch high chassis

Cray-1 Architecture (cont.)

- Memory (16 banks, 72 modules / bank)
  - 64 modules = 1 bit in 64 bit word
  - 8 modules = check byte for single-bit error correction, double bit error detection
- Functional units
  - 12 pipelined functional units in 4 groups: address, scalar, vector, and floating point
  - Scalar add = 3 cycles, vector add = 3 cycles, floating-point add = 6 cycles, floating-point multiply = 7 cycles, reciprocal approximation = 14 cycles
- Instruction formats
  - Either one or two 16-bit “parcels”
  - Arithmetic and logical instructions operate on 3 registers
  - Read & store instructions access memory

Cray-1 Registers

- Registers
  - 8 address registers (A), 64 address-save registers (B), 8 scalar registers (S), 64 scalar-save registers (T), & 8 64-word vector registers (V)
- 8 24-bit address registers (A)
  - Used as address registers for memory references and as index registers
  - Index the base register for scalar memory references, provide base address and index for vector memory references
  - 24-bit integer address functional units (add, multiply) operate on A data
- 64 24-bit address-save registers (B)
  - Used to store contents of A registers

Cray-1 Registers (cont.)

- 8 64-bit scalar registers (S)
  - Used in scalar operations
  - 64-bit integer scalar functional units (add, shift, logical, population/leading zero count) operate on S data
- 64 64-bit scalar-save registers (T)
  - Used to store contents of S registers, typically intermediate results of complex computations
- 8 64-element vector registers (V)
  - Each element is 64 bits wide
  - Each register can contain a vector of data (row of a matrix, etc.)
  - Vector Mask register (VM) controls elements to be accessed, Vector Length register (VL) specifies number of elements to be processed

Vector Arithmetic

- First, consider a vector on a SISD (non-parallel) machine
  - Vectors A, B, and C are each one-dimensional arrays of 10 integers
  - To add each corresponding value from A and B, storing the sum in C, would require at least 4 cycles, 40 cycles overall
  - If the CPU is a *vector processor*, loading, adding, and storing gets pipelined, so after a few cycles, a new value get stored into C each cycle, for 12 cycles overall, speedup of 40/12 = 3.33
  - The longer the vector, the more speedup
- Now consider a vector on a SIMD machine — each processor can do this vector processing in parallel
  - 64 processors => speedup of 213 over original computation!

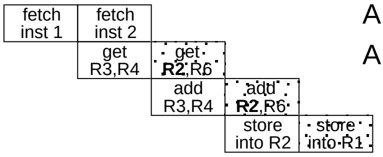
Chaining

- Vector operation operates on either two vector registers, or one vector register and one scalar register
- Parallel vector operations may be processed two ways:
  - Using different functional units and vector registers, or
  - By *chaining* — using the result stream from one vector register simultaneously as the operand set for another operation in a different functional unit
    - Intermediate results do not have to be stored in memory, and can even be used before a particular vector operation has finished
    - Similar to data forwarding in the IBM 360's pipeline

Handling Data Hazards

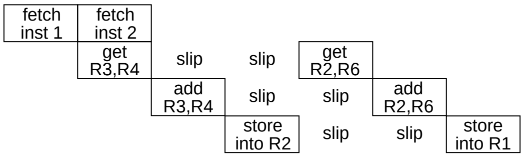
- *Write / read data hazard* example:

ADD R2, R3, R4
ADD R1, R2, R6



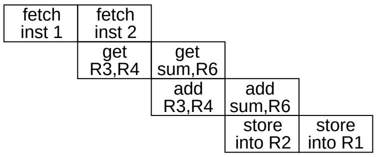
fetch inst 1	fetch inst 2		
	get R3,R4	get **R2**,R6	
		add R3,R4	add **R2**,R6
			store into R2	store into R1

- Can be avoided with *register interlocks*



fetch inst 1	fetch inst 2					
	get R3,R4	slip	slip	get R2,R6		
		add R3,R4	slip	slip	add R2,R6	
			store into R2	slip	slip	store into R1

- Can also be avoided with *data forwarding*



fetch inst 1	fetch inst 2		
	get R3,R4	get sum,R6	
		add R3,R4	add sum,R6
			store into R2	store into R1

Handling Data Hazards (cont.)

- Register interlocks
  - An instruction gets *blocked* until all its source registers are loaded with the appropriate values by earlier instructions
  - A "valid / invalid" bit is associated with each register
    - During decode stage, destination register is set to invalid (it will change)
    - Decode stage blocks until all its source (and destination) registers are valid
    - Store stage sets destination register to valid
- Data forwarding
  - Output of ALU is connected directly to ALU input buses
  - Result of an ALU operation is now available immediately to later instructions (i.e., even before it gets stored in its destination register)

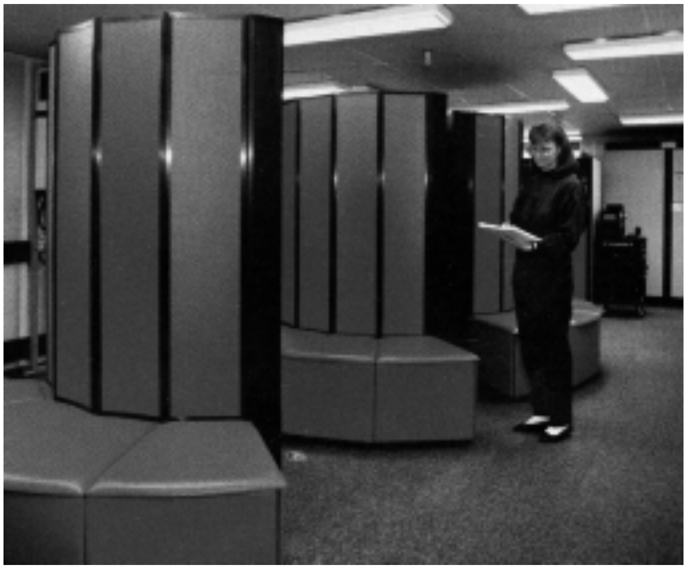
Miscellaneous

- Evolution
  - Seymour Cray was a founder of Control Data Corp. (CDC) and principal architect of CDC 1604 (non-vector machines)
  - 8600 at was to be made of tightly-coupled multiprocessors; it was cancelled so Cray left to form Cray Research
- Software
  - Cray Operating System (COS) — up to 63 jobs in a multiprog. environment
  - Cray Fortran Compiler (CFC) — optimizes Fortran IV (1966) for the Cray-1
    - Automatically vectorizes many loops that manipulate arrays
- Front-end computer
  - Any computer, such as a Data General Eclipse or IBM 370/168

Cray X-MP, Y-MP, and {CJT}90

- At Cray Research, Steve Chen continued to update the Cray-1, producing…
- X-MP
  - 8.5 ns clock (Cray-1 was 12.5 ns)
  - First multiprocessor supercomputer
    - 4 vector units with scatter / gather
- Y-MP
  - 32-bit addressing (X-MP is 24-bit)
  - 6 ns clock
  - 8 vector units
- C90, J90 (1994), T90
  - J90 built in CMOS, T90 from ECL (faster)
  - Up to 16 (C90) or 32 (J90/T90) processors, with one multiply and one add vector pipeline per CPU

Cray X-MP @ National Supercomputer Center in Sweden



Cray-2 & Cray-3

- At Cray Research, Steve Chen continued to update the Cray-1 with improved technologies: X-MP, Y-MP, etc.
- Seymour Cray developed Cray-2 in 1985
  - 4-processor multiprocessor with vectors
  - DRAM memory (instead of SRAM), highly interleaved since DRAM is slower
  - Whole machine immersed in Fluorinert (artificial blood substitute)
  - 4.1 ns cycle time (3x faster than Cray-1)
  - Spun off to Cray Computer in 1989
- Seymour Cray developed Cray-3 in 1993
  - Replace the “C” shape with a cube so all signals take same time to travel
  - Supposed to have 16 processors, had 1 with a 2 ns cycle time