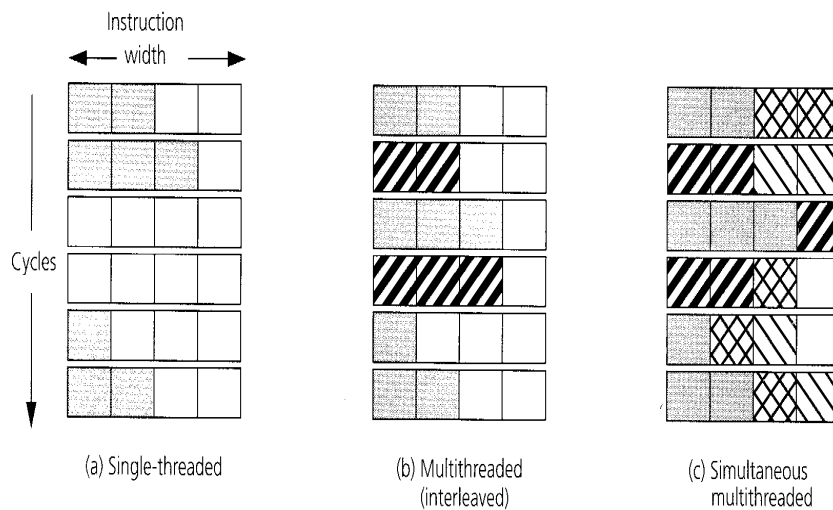


**CS/ECE 757: Advanced Computer Architecture II
(Parallel Computer Architecture)**

Processor Multithreading: HEP

**Copyright 2003 J. E. Smith
University of Wisconsin-Madison**

Multithreading



Multi-Threaded MPs

- **History: CDC6600 PPs and Denelcor HEP**
- **HEP**
 - General purpose scientific

CDC 6600 Peripheral Processors

- **Intended to perform OS and I/O functions**
- **Used "barrel and slot"**
 - register state is arranged around a barrel
 - one set of ALU and memory hardware accessed through slot in barrel
 - slot in barrel rotates a position each cycle
- **Could be used as stand-alone "MP"**
- **Similar method later used in IBM Channels**

Denelcor HEP

- **General purpose scientific computer**
- **Organized as an MP**
 - Up to 16 processors
 - Each processor is multithreaded
 - Up to 128 memory modules
 - Up to 4 I/O cache modules
 - Three-input switches and chaotic routing

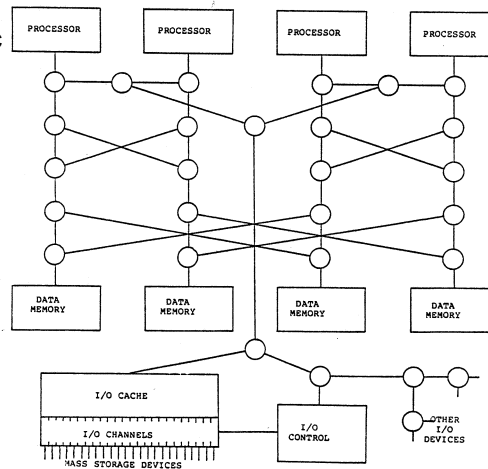


Figure 1. A typical HEP system

(C) 2003 J. E. Smith

ECE/CS 757

5

Processor Organization

- **Multiple contexts (threads) are supported;**
 - each with a PSW (program status word)
- **PSWs circulate in a control loop**
 - control and data loops pipelined 8 deep
 - PSW in control loop can circulate no faster than data in data loop
 - PSW at queue head fetches and starts execution of next instruction
- **Clock period: 100 ns**
 - 8 PSWs in control loop => 10 MIPS
 - Maximum perf. per thread => 1.25 MIPS
 - (And they tried to sell this as a supercomputer)

(C) 2003 J. E. Smith

ECE/CS 757

6

Processor Organization

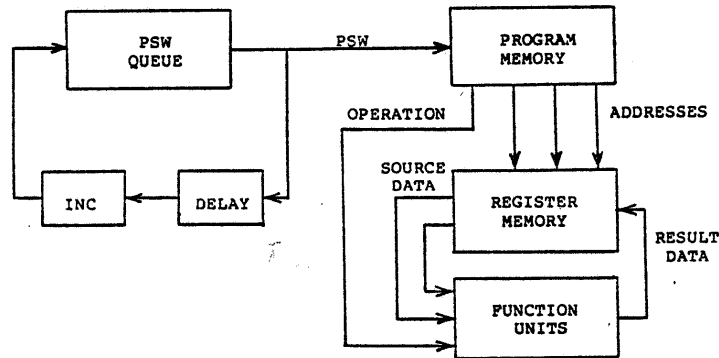


Figure 2. Simplified HEP processor

Processor, contd.

- **Address space: 32K to 1Mwords (64 bits)**
- **64 bit instructions**
- **2048 GP registers + 4096 constants**
- **Memory operation**
 - Loads and stores performed by scheduler functional unit (SFU)
 - SFU builds network packet and sends it into switch
 - PSW is removed from control loop and placed in SFU queue
 - PSW is placed back into control loop following memory response
- **Special operations**
 - control instructions to create/terminate threads
 - full/empty bits in memory and registers
 - » busy wait on empty/full operands

Parallelism

- **Independent programs**
- **Multiple threads with the same protection domain**
(memory and register space)

Switch

- **Packet switched**
- **3 bi-directional ports per switch**
 - every cycle, take in 3 packets, send out 3 packets
- **"Hot Potato" routing**
 - Form of adaptive routing
 - Do not enqueue on a port conflict
 - » Send anyway on another port and raise priority
 - At top priority (15) traverse a circuit through the net

HEP Programming

- **CREATE/RESUME**
 - analogous to call/return
- **Asynchronous variables: empty/full state**

HEP Programming Example

```

      :
      PURGE $IP, $NP
      $NP = NPROCS
      DO 10 I = 2, NPROCS
      $IP = I-1
      CREATE S($IP,$NP)
10    CONTINUE
      $IP = NPROCS
      CALL S($IP,$NP)
C     WAIT FOR ALL PROCESSES TO FINISH
20    N = $NP
      $NP = N
      IF (N .NE. 0) GO TO 20
      :
      SUBROUTINE S($IP,$NP)
      MYNUM = $IP
      :
      $NP = $NP-1
      RETURN
      END
      :

```

Figure 4. HEP Fortran example

HEP Programming Example

```
C      REPLACE EACH ELEMENT OF THE VECTOR X
C      BY THE SUM OF THE ELEMENTS OF X
C      USING THE INITIALLY EMPTY ARRAY SA.
C      P = 2**L PROCESSES EXECUTE THIS PROGRAM.
C      THE PROCESS IDENTIFER IS I.
C      DIMENSION X(P), A(P,L)
C      JPOW = 2
C      JPOW IS 2 TO THE J POWER
C      DO 10 J = 1,L
C      COMPUTE THE PROCESS K = (I-1) EXOR(JPOW/2)+1
C      WITH WHICH THIS PROCESS WILL EXCHANGE DATA
C      K = ((I-1)/JPOW)*JPOW + MOD (I-1 + JPOW/2,JPOW)+1
C      JPOW = JPOW*2
C      NOW EXCHANGE DATA AND ACCUMULATE
C      SA(K,J) = X(I)
C      X(I) = X(I) + SA(I,J)
10    CONTINUE
```

Figure 5. Summation by network simula

HEP Programming Example

```
DO 10 I = J, K, L
:
:
10 CONTINUE
```

Figure 6. A DO loop

HEP Programming Example

```
      PURGE $IV  
      $IV = J  
C     CREATE ANY NUMBER OF PROCESSES EXECUTING  
1     I = $IV  
C     THIS PROCESS HAS SEIZED AN ITERATION INDEX  
C     LET ANOTHER PROCESS OBTAIN THE NEXT INDEX  
      $IV = I + L  
C     TERMINATE IF THROUGH  
      IF (I .GT. K) RETURN  
      :  
      GO TO 1
```

Figure 7. A self-scheduled loop