Computer Architecture Basics

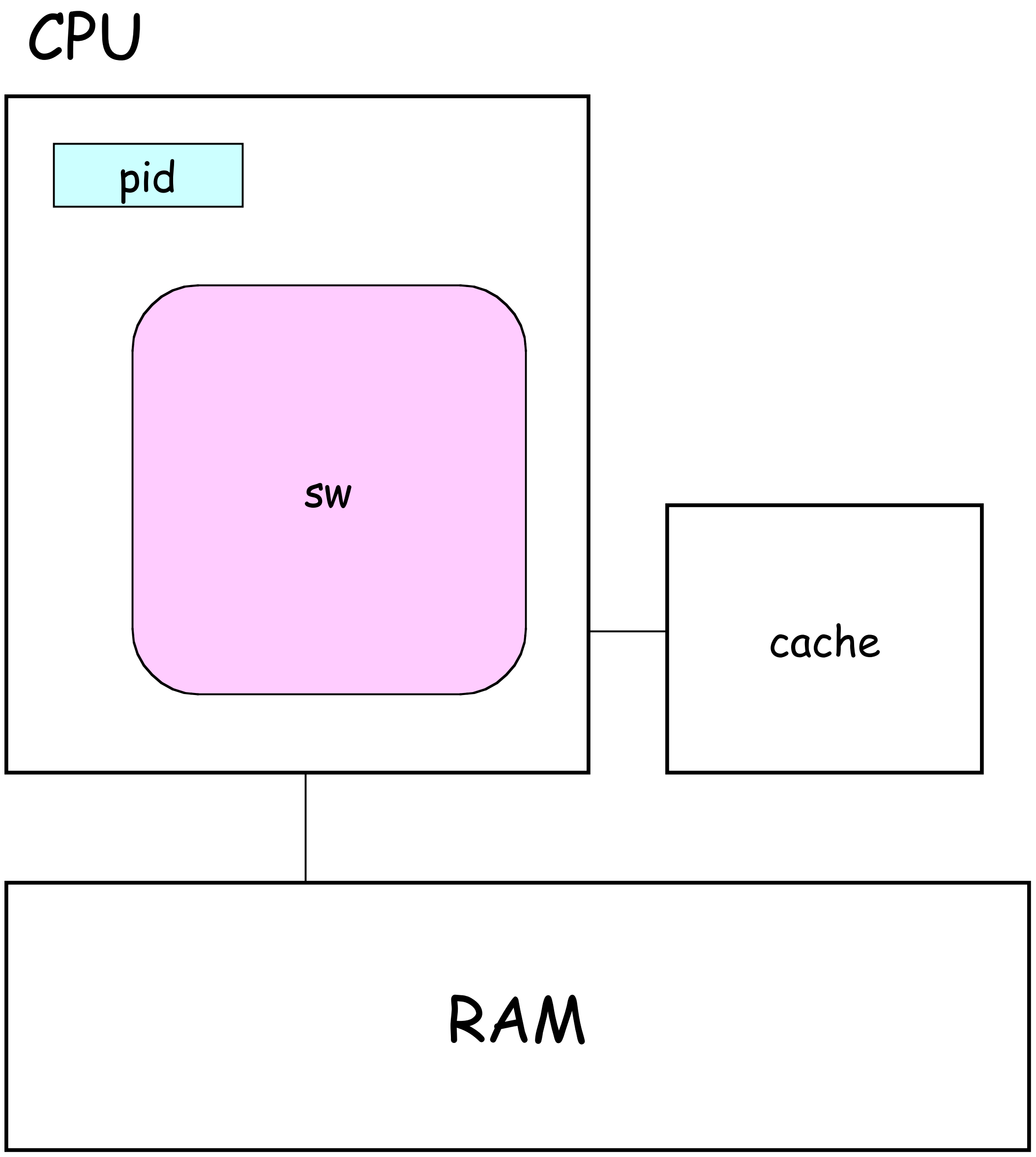
P. J. Denning

For CS471/CS571

Elements

Interface with OS software and functions

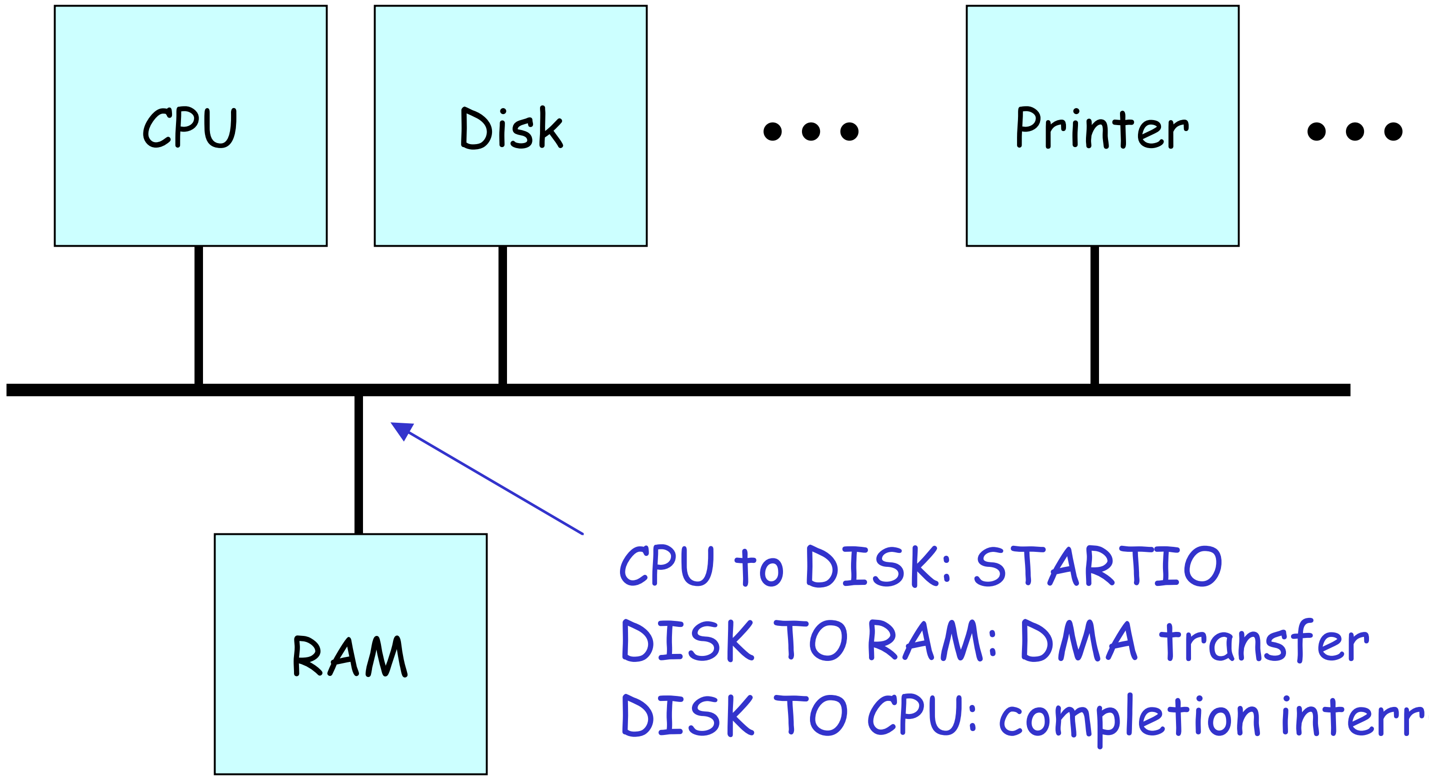
- Basic CPU
- Basic I/O
- Cache
- Procedures
- Memory protection
- Privilege modes
- Interrupts
- Statewords
- Processes


CPU
pid
sw
cache
RAM

Basic CPU -- Stateword

- Arithmetic registers
- Address registers
- Instruction pointer register
- Supervisor/User mode flipflop
- Interrupt mask register
- Stack pointer register
- Stack environment registers
- Timer alarm clock

Basic I/O



CPU to DISK: STARTIO
DISK TO RAM: DMA transfer
DISK TO CPU: completion interrupt

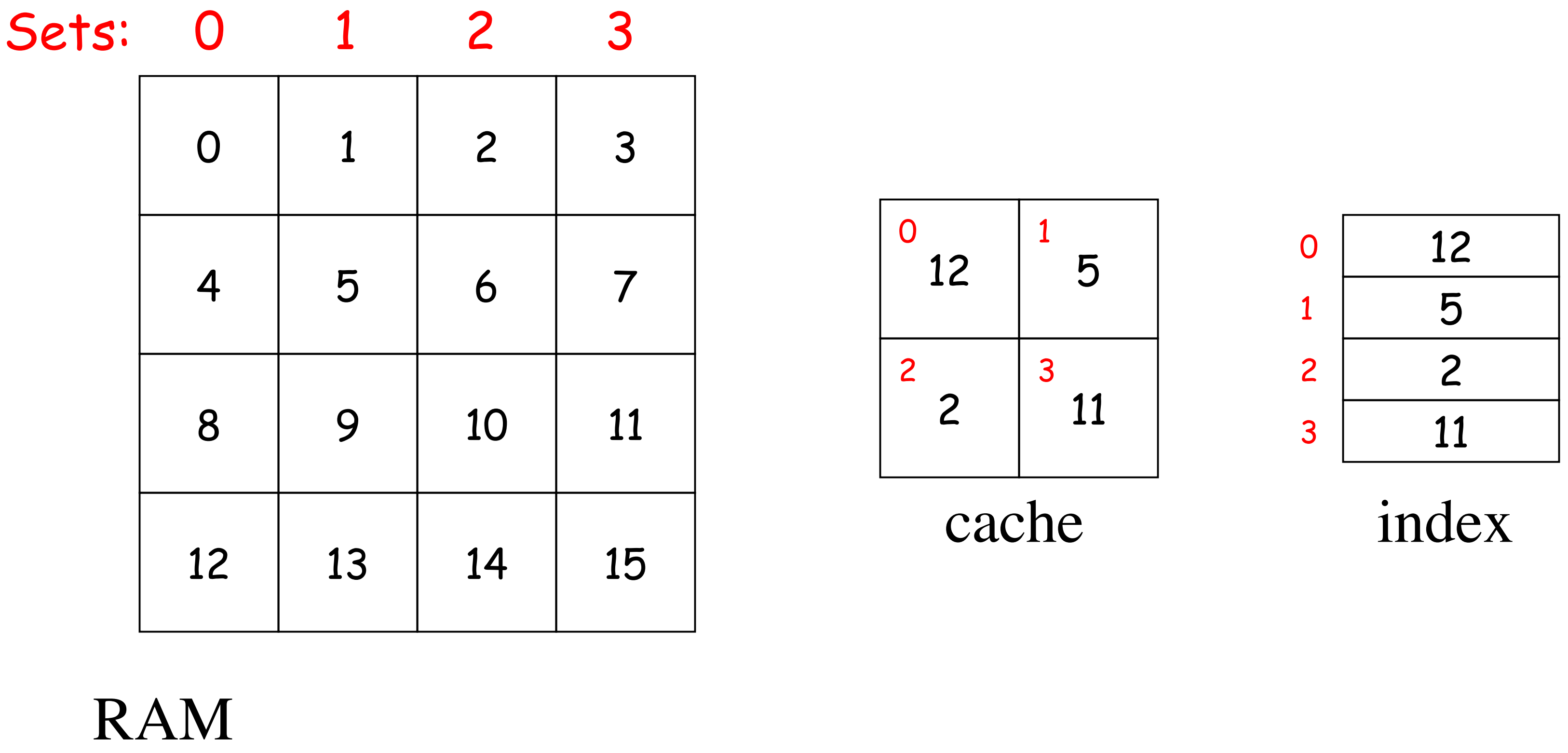
Basic I/O

- STARTIO(i, j, a, b, rw, size)
  - Method 1: Package (a,b,rw,size) in packet from sender j to device i
  - Method 2: Place (a,b,rw,size) in device block in memory --- memory mapped I/O
- Transfer via Direct memory access (DMA)
- Finish with completion interrupt

Cache

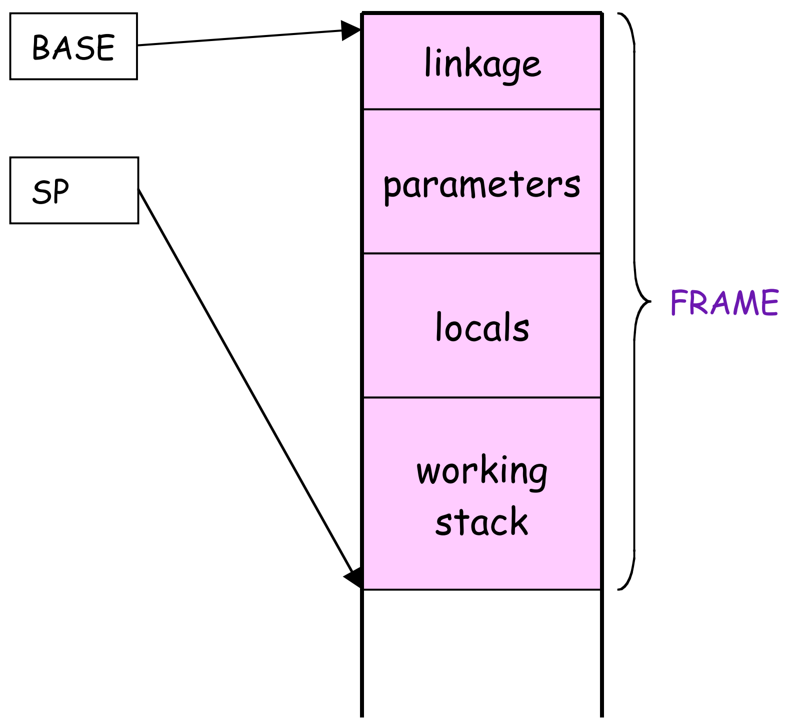
Hold subset of RAM contents in fast local-access store

- Blocks of memory all of same size
  - RAM block = page
  - Cache block = slot
- RAM set = fixed set of pages mapping to a single slot; same number of pages in each set
- Cache holds one slot per set
- Index: table telling which page in each slot


Sets: 0 1 2 3
0 1 2 3
4 5 6 7
8 9 10 11
12 13 14 15
RAM
0 12
1 5
2 2
3 11
cache
0 12
1 5
2 2
3 11
index

$$\underbrace{T}_{\boxed{\text{effective access time}}} \quad = \quad \overbrace{(h)}^{\boxed{\text{hit probability}}}(\text{cache time}) + (1\text{-}h)(\text{RAM time})$$

Procedure Call Stack



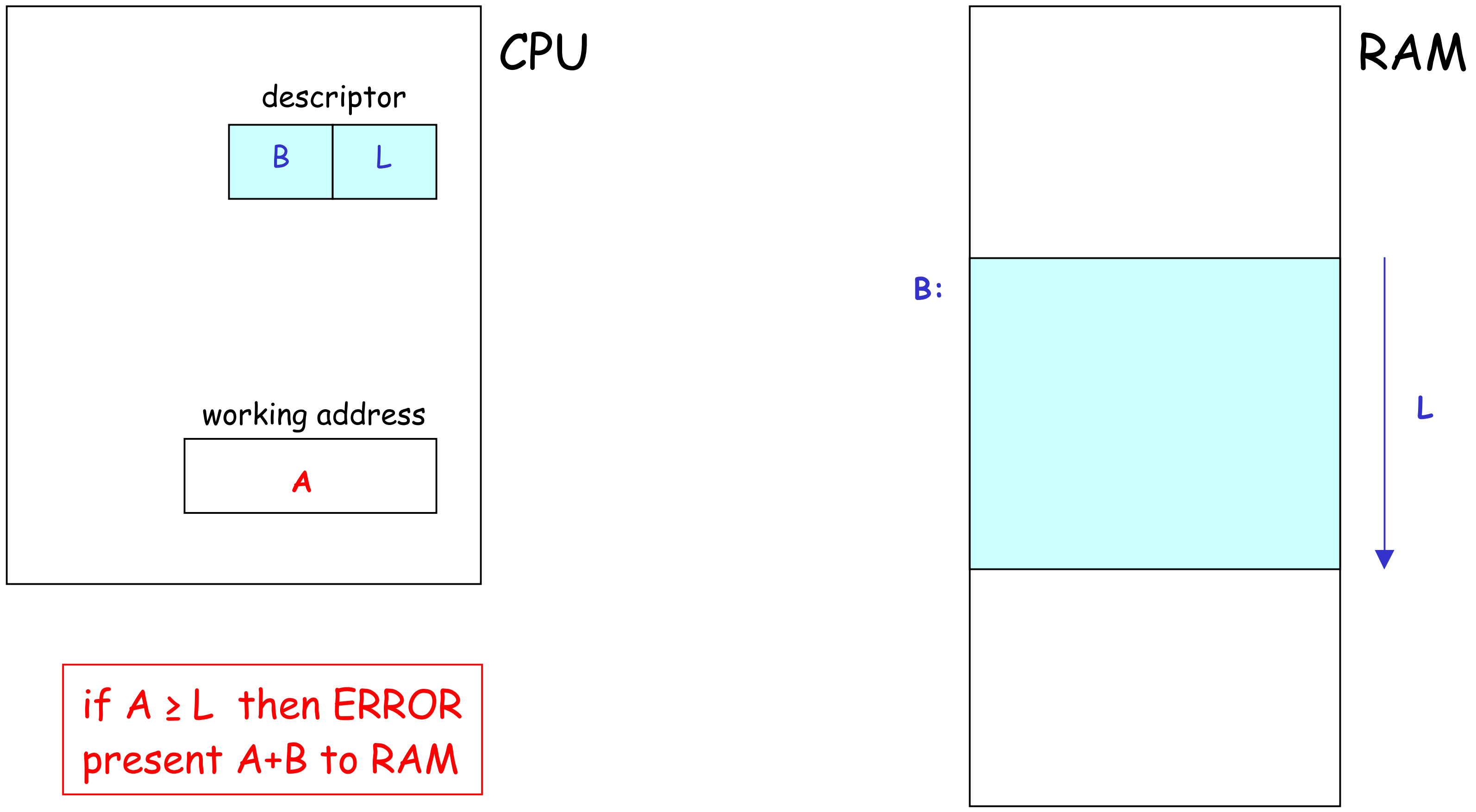
Call stack manages a set of frames (activation records) of called procedures on a stack.

CPU stack environment registers keep track of the stack.

Linkage section contains return information to restore caller's environment.

See separate presentation on procedures for more detail.

Memory Protection



if $A \geq L$ then ERROR
present A+B to RAM

Interrupts

handle =
entry point address
+ supervisor/user mode value
+ interrupt mask

CALL instruction uses handle to set IP, S, and M

- Event detectors
- Interrupt mask bits
- Interrupt number
- Interrupt vector = interrupt handler handles
- Interrupt handler
- Interrupt parameter (passed in register or atop stack)
- System Call instruction

Hardware Interrupt Dispatch

Interrupt signal

Select i = highest priority pending signal

CALL IH[i] (sets IP, S, M) and pass parameter

Statewords & Processes

- Process: abstraction of running program
- Process = dynamic trace of IP moving through program code
- Process = sequence of statewords of a running program
- Process must have address space = context of code, data, and stack

Statewords & Processes

- pid register in CPU = identity of process using CPU
- stateword of CPU = contents of all registers (except pid)
- PCB = process control block
  = sw + LINK
- Preset for processes 0, 1, 2, …, N

Statewords & Processes

- Ready list = FIFO of all CPU-ready pids
- RL = (h,t) descriptor linking ready PCB's
- SAVESW -- sw to PCB[pid]
- LOADSW -- PCB[RL.h].SW to CPU
- Context switch = (SAVESW, LOADSW)

Statewords & Processes

- Create process (FORK) = initialize free PCB with SW, link to RL tail
- Delete process (JOIN) = mark PCB as free
- FL = (h,t) descriptor linking all free PCB's
- Process 0 = idle process, always at end of RL if LINK 0 is endmarker

Simple Scheduling (Round Robin)

- Clock interrupt switches CPU to next ready process
- Q = time quantum = max time for CPU to run in one process
- Clock interrupt handler:

```
disable
SAVESW
link pid to RL tail
set alarm clock = Q
LOADSW
enable
return
```

A more detailed description of these elements of process management can be found in the slides about time sharing.