238P: Operating Systems

Lecture 3: PC Hardware

Anton Burtsev
January, 2018

Recap from last time

Xv6 system calls

fork() Create a process
exit() Terminate the current process
wait() Wait for a child process to exit
kill(pid) Terminate process pid
getpid() Return the current process's pid
sleep(n) Sleep for n clock ticks
exec(filename, *argv) Load a file and execute it
sbrk(n) Grow process's memory by n bytes
open(filename, flags) Open a file; the flags indicate read/write
read(fd, buf, n) Read n bytes from an open file into buf
write(fd, buf, n) Write n bytes to an open file
close(fd) Release open file fd
dup(fd) Duplicate fd
pipe(p) Create a pipe and return fd's in p
chdir(dirname) Change the current directory
mkdir(dirname) Create a new directory
mknod(name, major, minor) Create a device file
fstat(fd) Return info about an open file
link(f1, f2) Create another name (f2) for the file f1
unlink(filename) Remove a file

Processes

fork() Create a process

exit() Terminate the current process

wait() Wait for a child process to exit

kill(pid) Terminate process pid

getpid() Return the current process's pid

exec(filename, *argv) Load a file and execute it

Files

open(filename, flags) Open a file; the flags indicate read/write

read(fd, buf, n) Read n bytes from an open file into buf

write(fd, buf, n) Write n bytes to an open file

close(fd) Release open file fd

I/O redirection and interprocess communication

close(fd) Release open file fd

dup(fd) Duplicate fd

pipe(p) Create a pipe and return fd’s in p

Memory

`sbrk(n)` `Grow process's memory by n bytes`

Directories and named files

chdir(dirname) Change the current directory

mkdir(dirname) Create a new directory

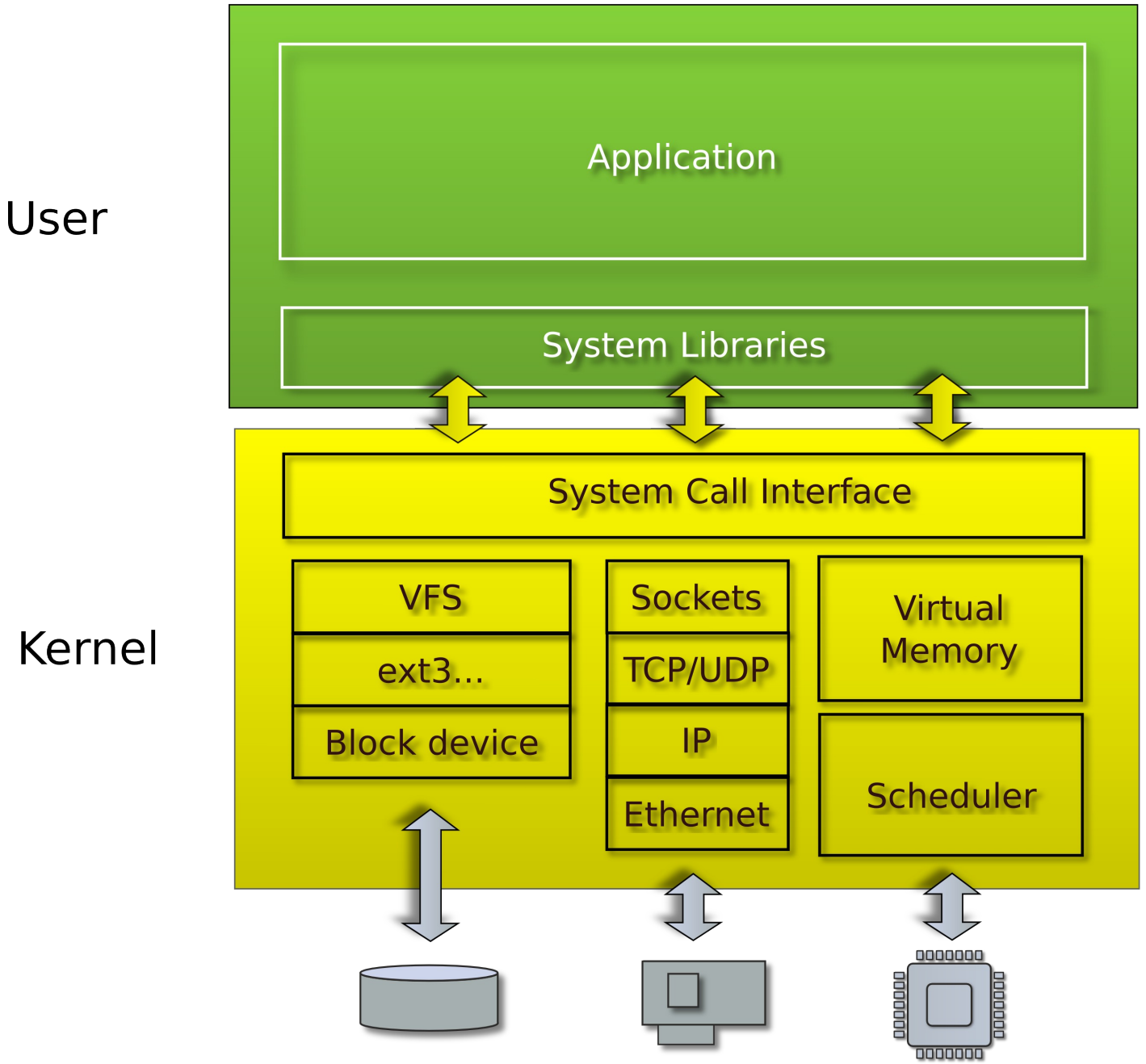
mknod(name, major, minor) Create a device file

fstat(fd) Return info about an open file

link(f1, f2) Create another name (f2) for the file f1

unlink(filename) Remove a file

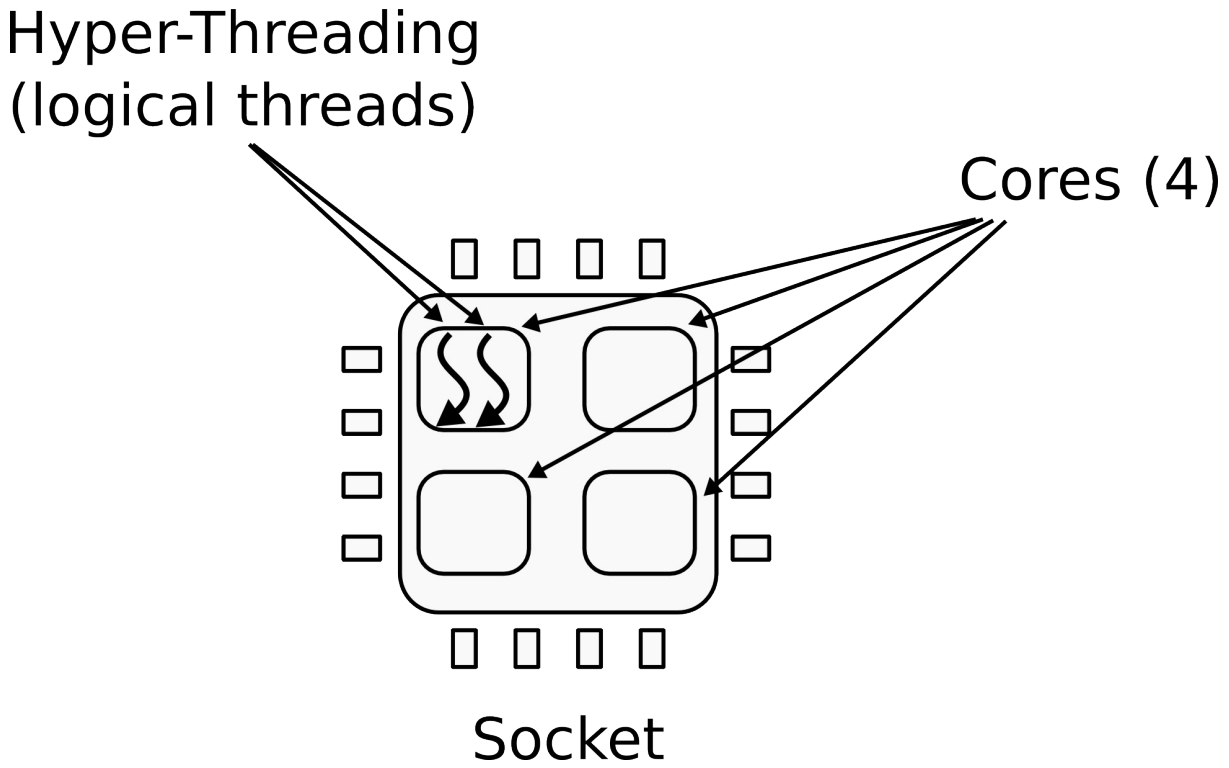
OS implements this interface



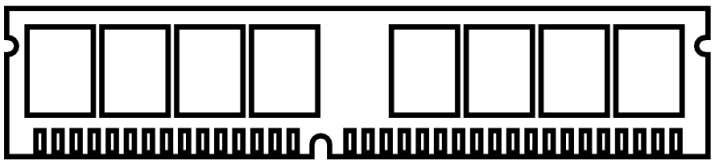
PC Hardware

CPU

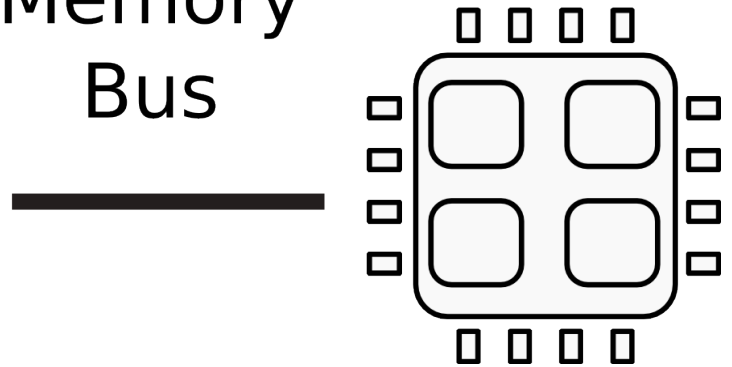
- 1 CPU socket
  - 4 cores
  - 2 logical (HT) threads each

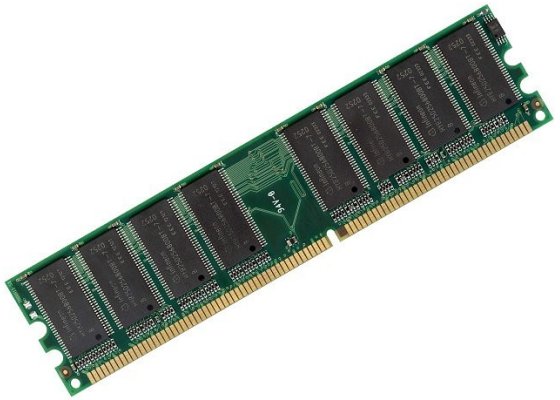


Memory



Memory
Bus





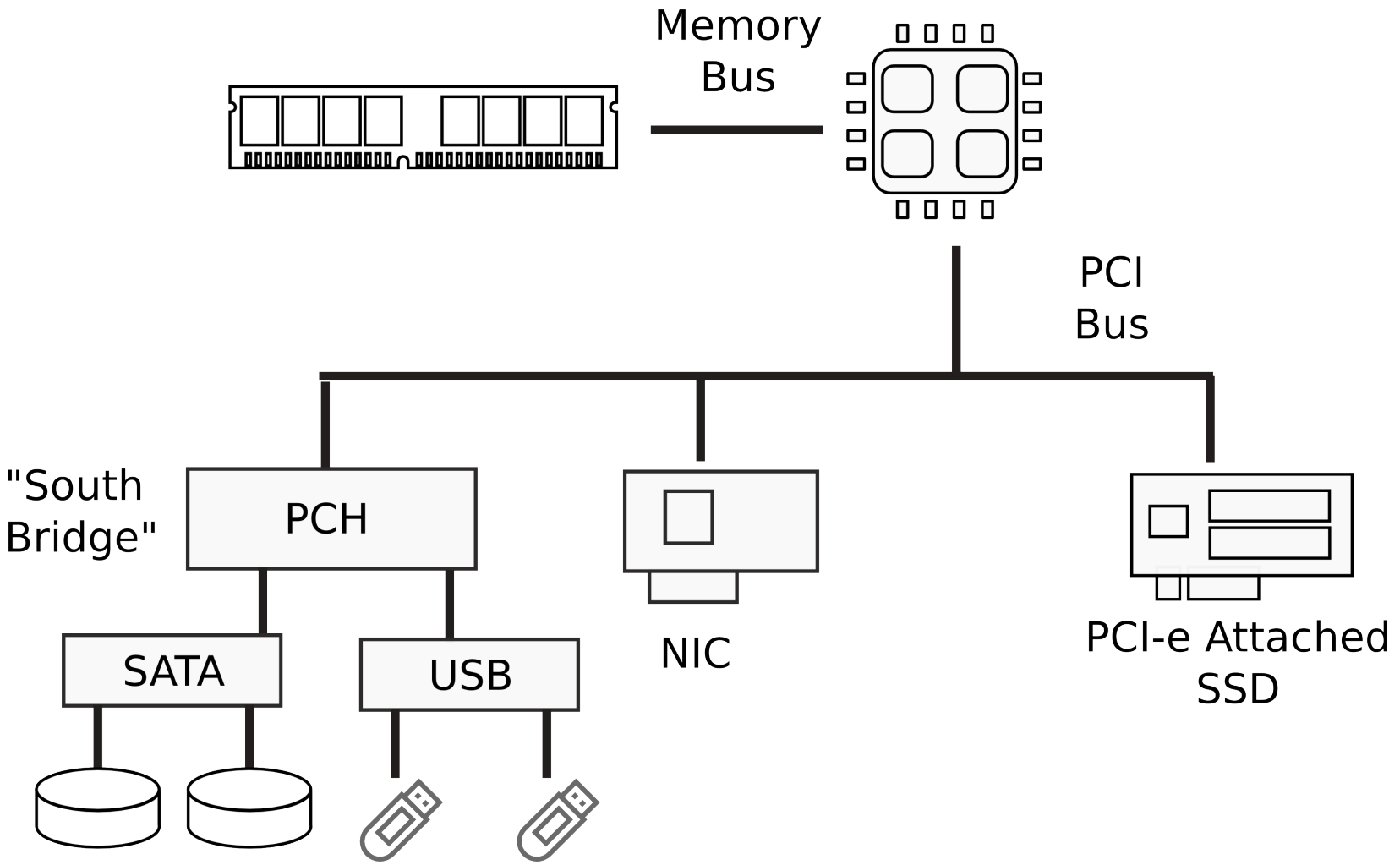
Memory abstraction

WRITE(*addr*, *value*) → ∅ Store *value* in the storage cell identified by *addr*.
READ(*addr*) → *value* Return the *value* argument to the most recent WRITE call referencing *addr*.

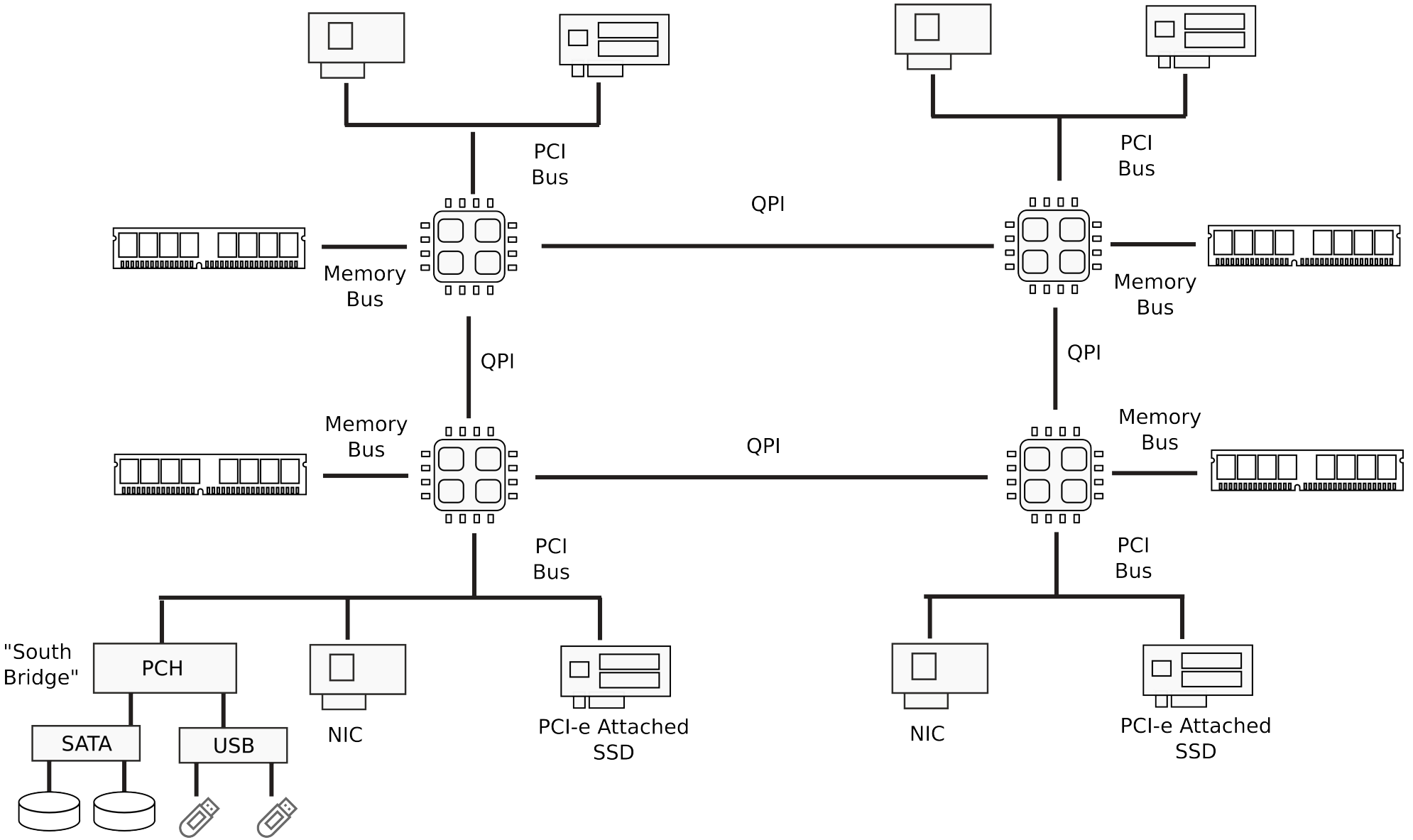
- x86 assembly example:

```
mov eax, [ebx] ; Move 4 bytes in memory at the address contained in EBX into EAX
```

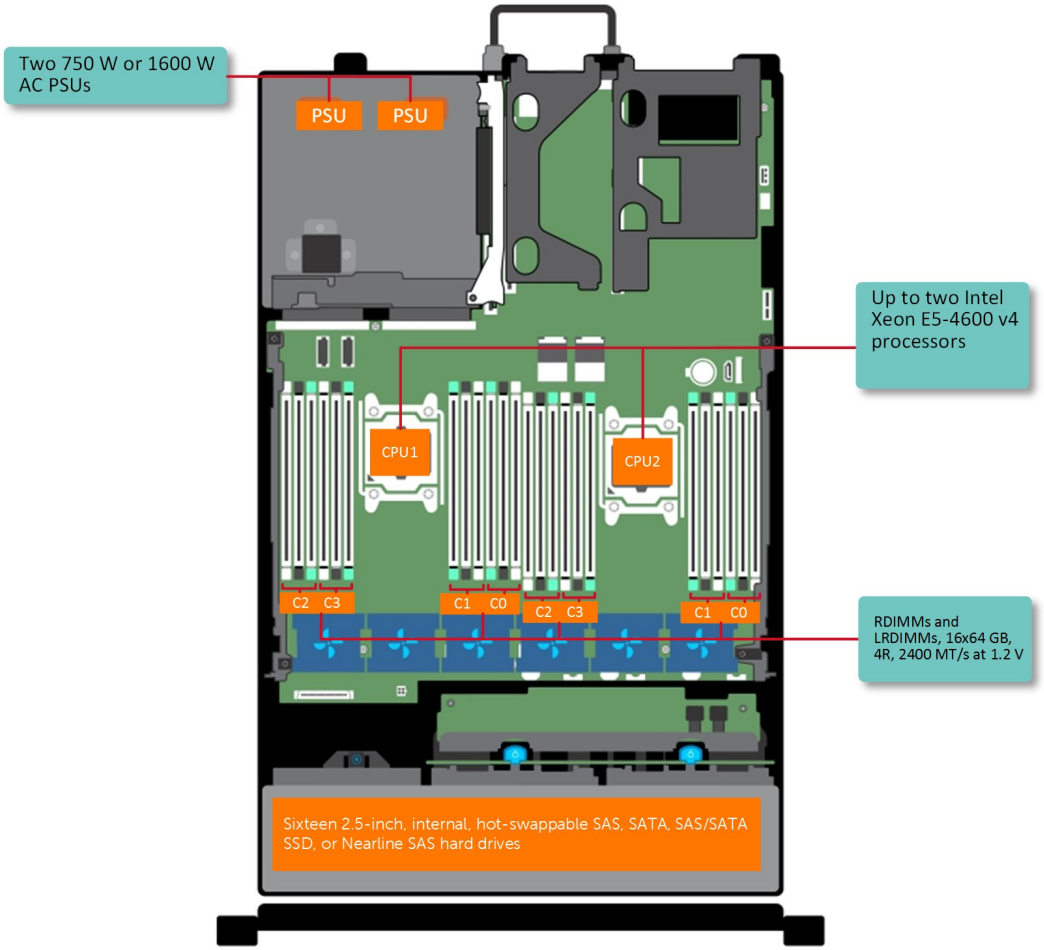
I/O Devices

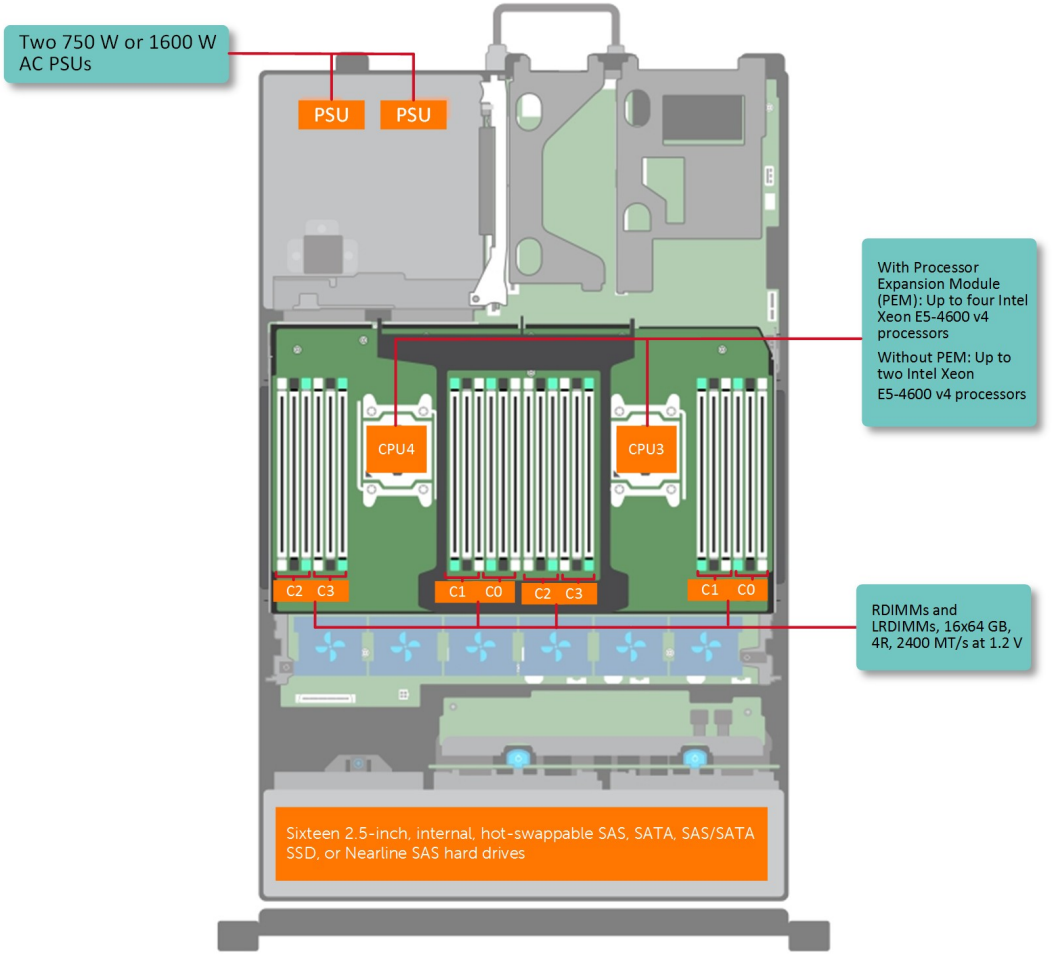

Memory
Bus
PCI
Bus
"South
Bridge"
PCH
SATA
USB
NIC
PCI-e Attached
SSD

Multi-socket machines



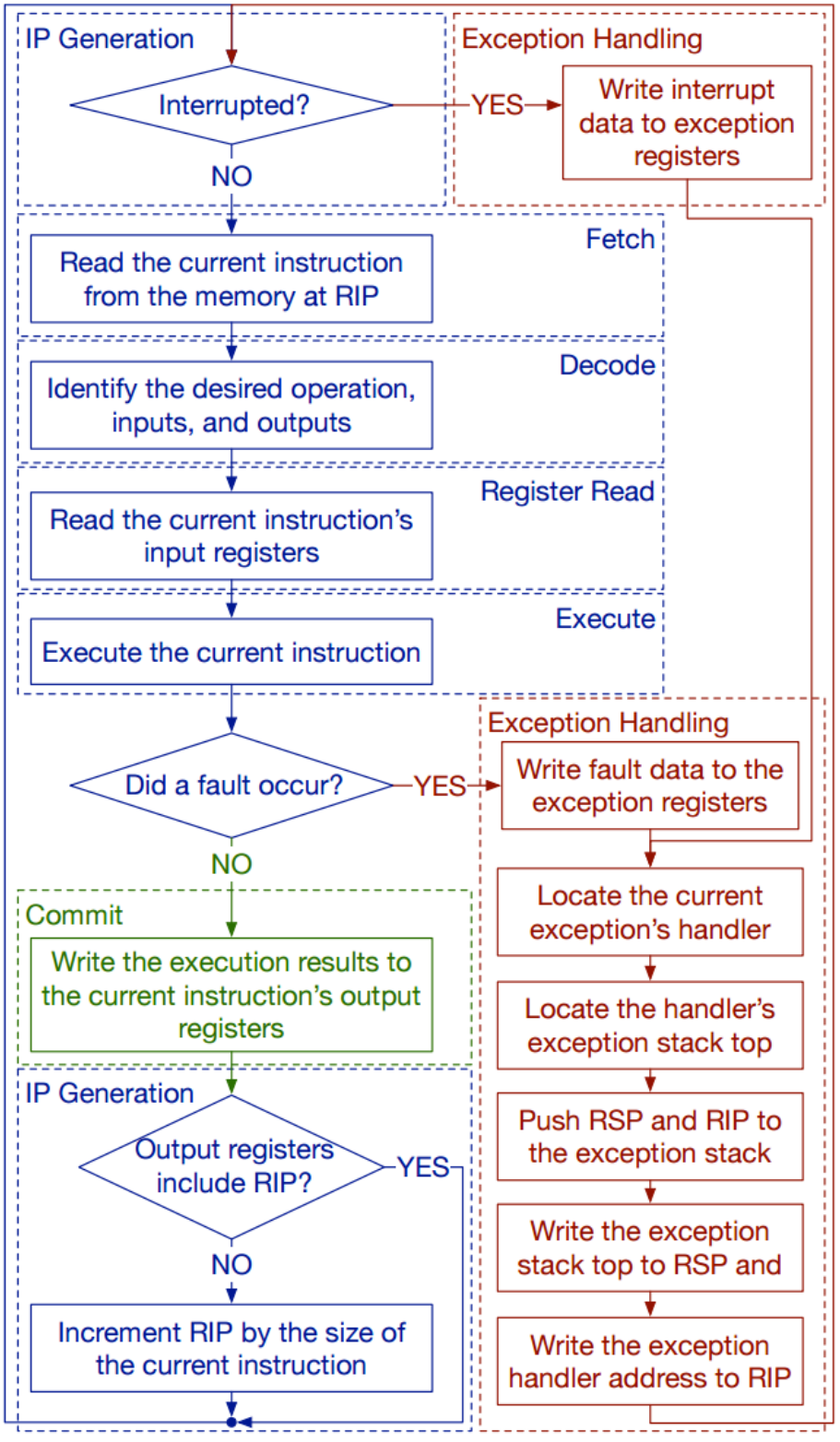
Dell R830 4-socket server





Dell Poweredge R830 System Server with 2 sockets on the main floor and 2 sockets on the expansion

http://www.dell.com/support/manuals/us/en/19/poweredge-r830/r830_om/supported-configurations-for-the-poweredge-r830-system?guid=guid-01303b2b-f884-4435-b4e2-57bec2ce225a&lang=en-us

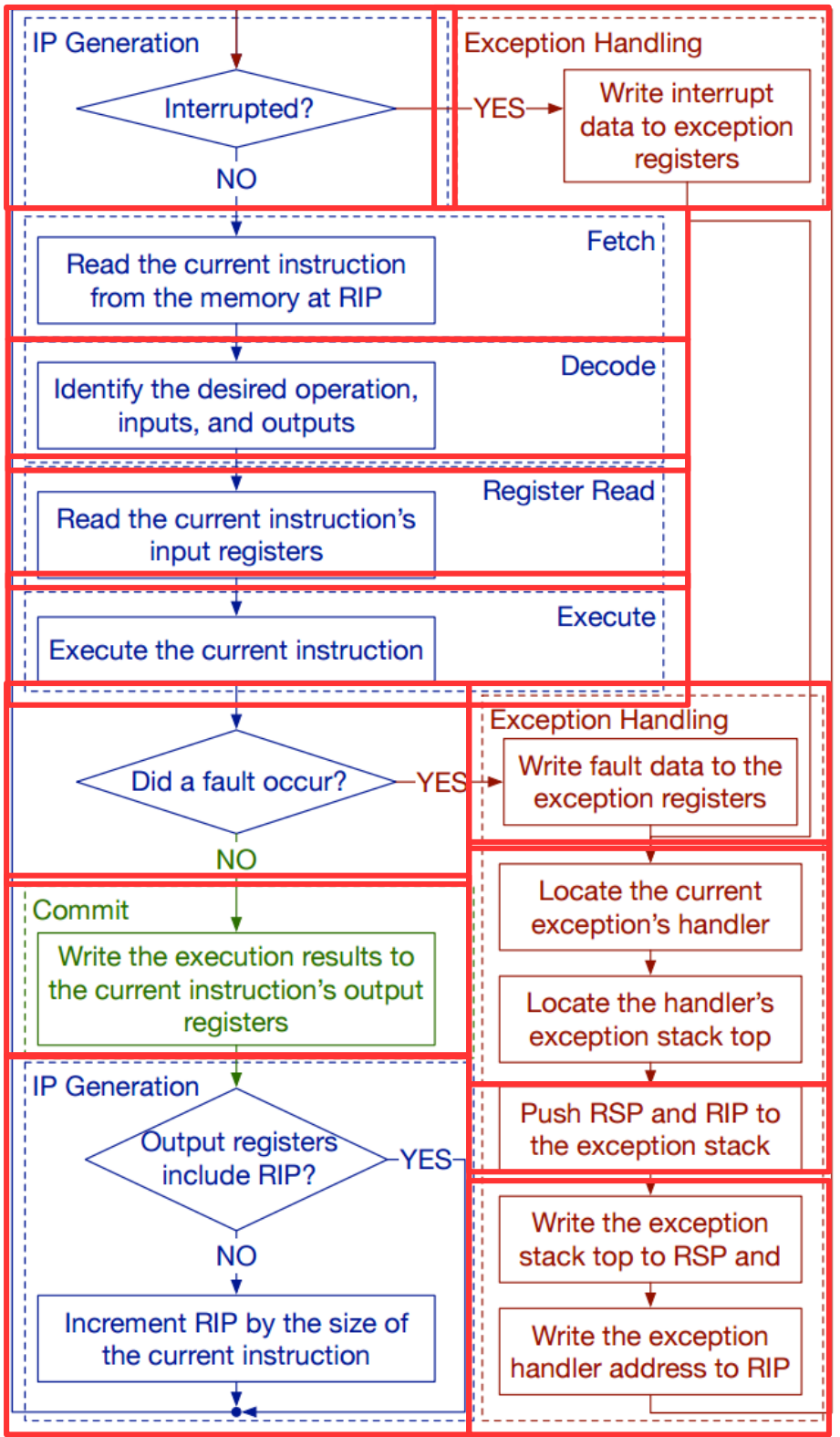


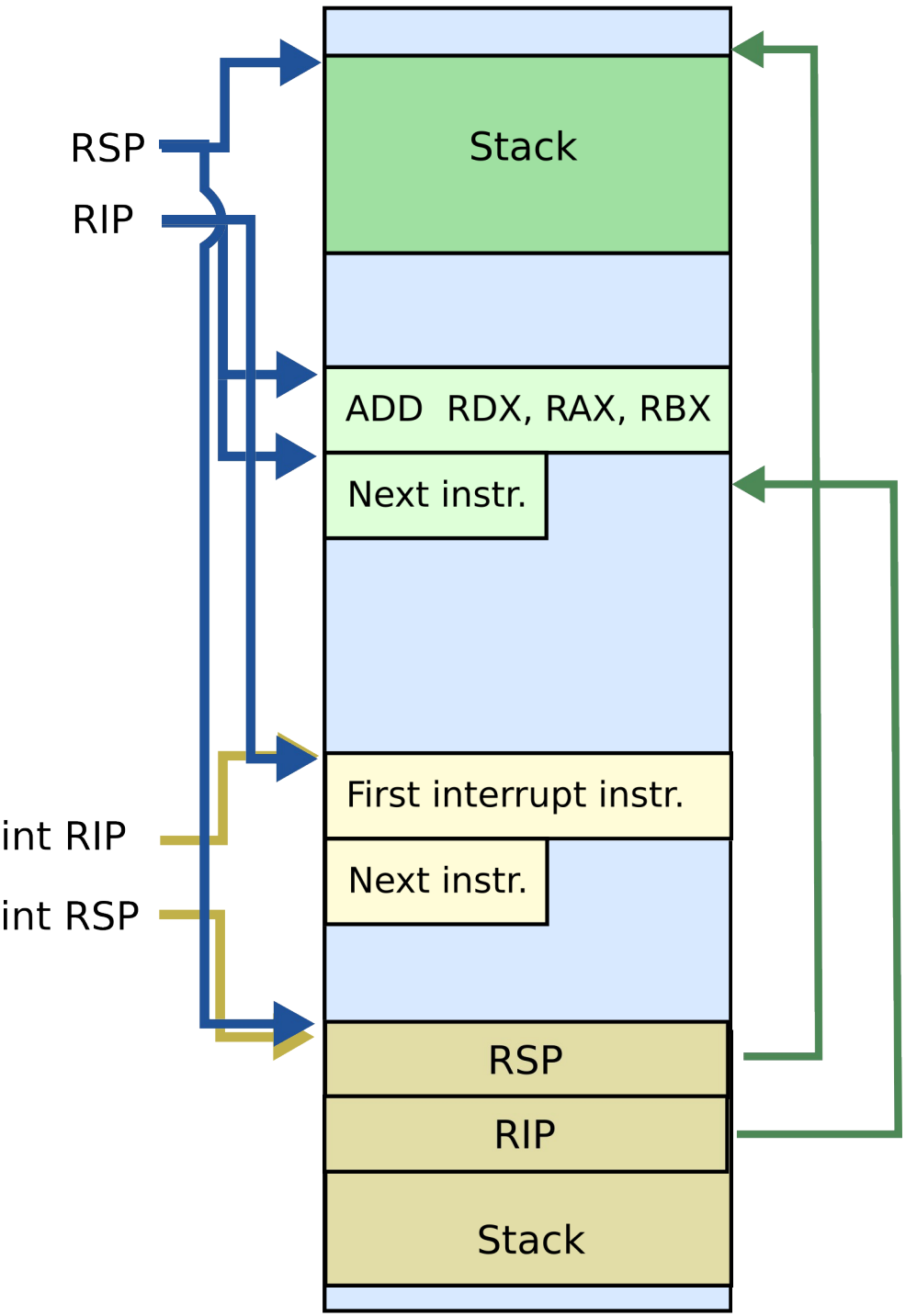
CPU execution loop

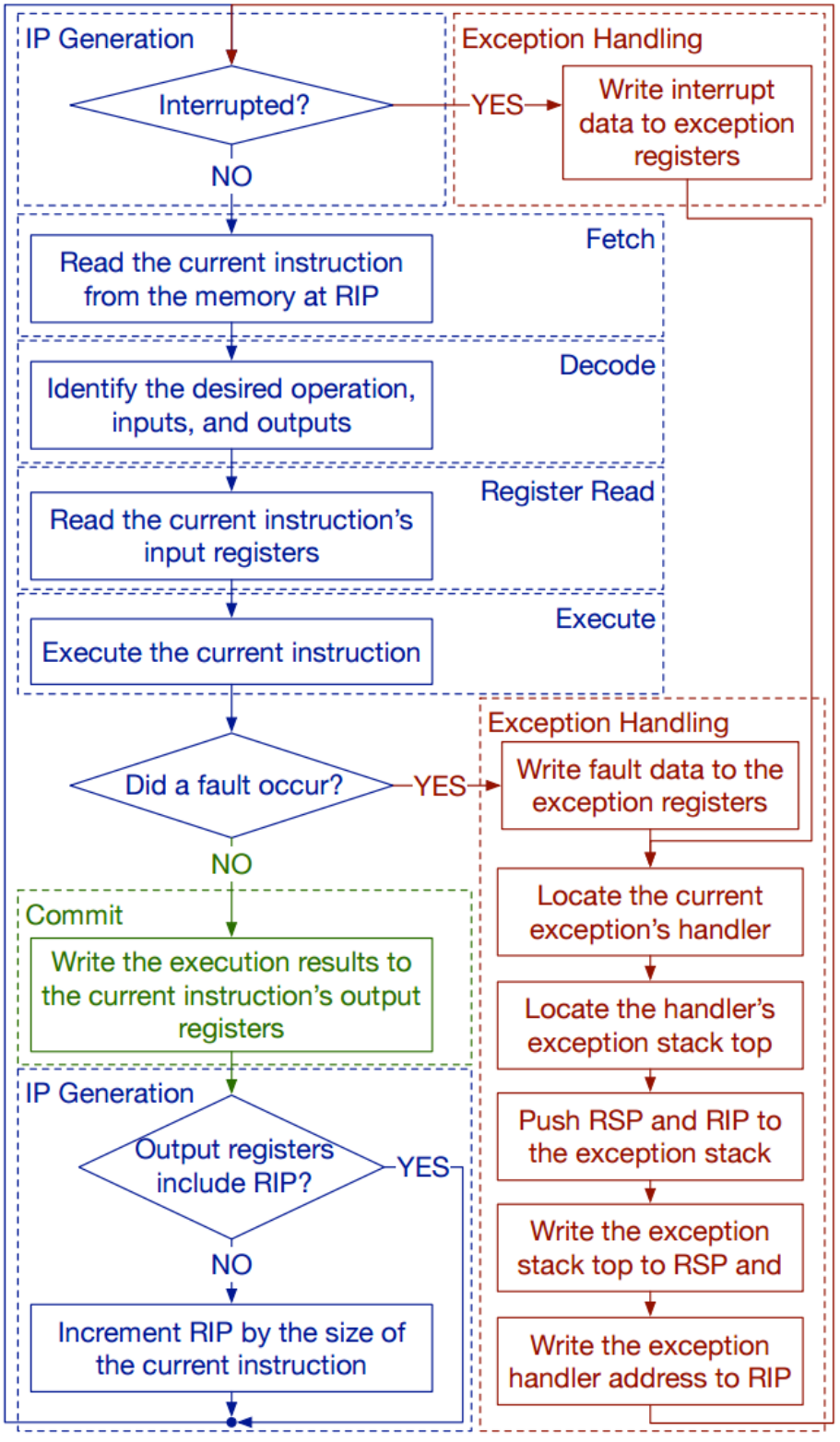
- CPU repeatedly reads instructions from memory
- Executes them
- Example

```
ADD EDX, EAX, EBX

// EDX = EAX + EBX
```


IP Generation
Interrupted?
YES
NO
Exception Handling
Write interrupt data to exception registers
Fetch
Read the current instruction from the memory at RIP
Decode
Identify the desired operation, inputs, and outputs
Register Read
Read the current instruction's input registers
Execute
Execute the current instruction
Did a fault occur?
YES
NO
Exception Handling
Write fault data to the exception registers
Locate the current exception's handler
Locate the handler's exception stack top
Push RSP and RIP to the exception stack
Write the exception stack top to RSP and
Write the exception handler address to RIP
Commit
Write the execution results to the current instruction's output registers
IP Generation
Output registers include RIP?
YES
NO
Increment RIP by the size of the current instruction


RSP
RIP
int RIP
int RSP
Stack
ADD RDX, RAX, RBX
Next instr.
First interrupt instr.
Next instr.
RSP
RIP
Stack



CPU execution loop

- Fault
  - Instruction's preconditions are not met
- Examples
  - Division by zero
  - Page not mapped

Memory hierarchy

Questions?