CS 33: Introduction to Computer Organization

TA: Aalisha Dalal
LA: Jonathan Myong
Office Hours: Friday, 9:30-11:30AM

Outline

- **Locking mechanism (quick recap)**
- **Exceptions**
- **Linking**
- **Virtual Memory**
- **Worksheet Problems**

Locking Mechanism

What happens if threads are at this point at the same time?

Thread A	Thread B
`lock (lock_a);`	`lock (lock_b);`
`a += 5;`	`b += 5;`
`lock (lock_b);`	`lock (lock_a);`
`b += 7;`	`a += 7;`
`a += b;`	`a += b;`
`unlock (lock_b);`	`unlock (lock_a);`
`a += 11;`	`b += 11;`
`unlock (lock_a);`	`unlock (lock_b);`

If there is an issue, how can we solve it?

Solution to Locking Problem

Threads must lock lock_a before lock_b

Thread A

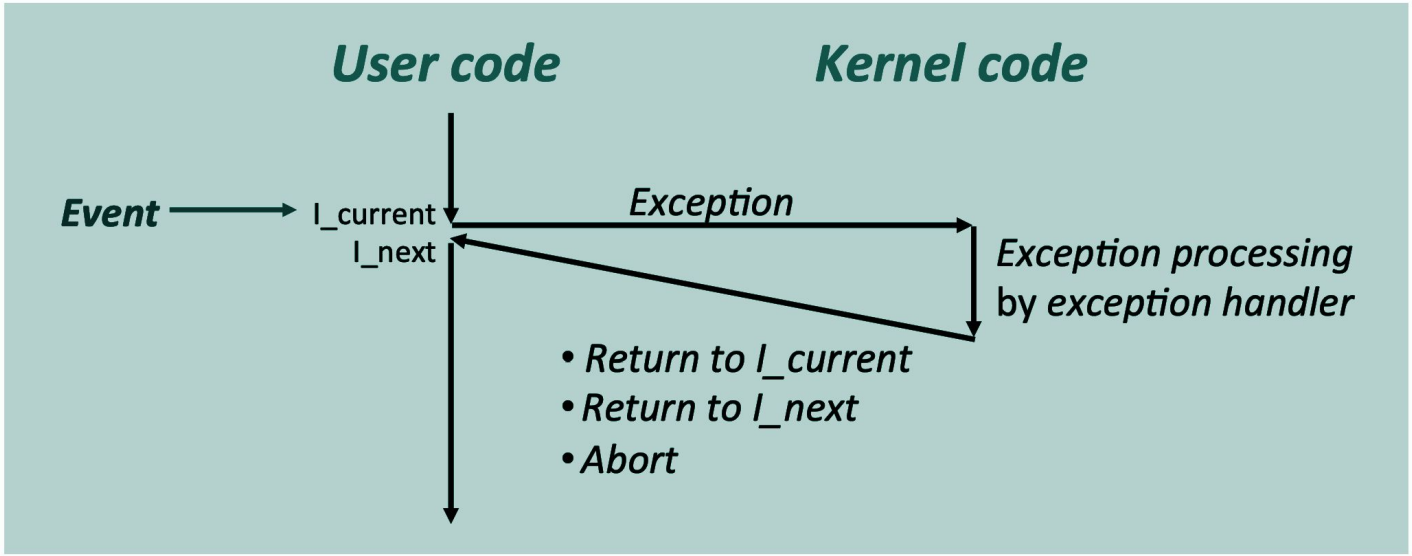
```
lock (lock_a);
a += 5;
lock (lock_b);
b += 7;
a += b;
unlock (lock_b);
a += 11;
unlock (lock_a);
```

Thread B

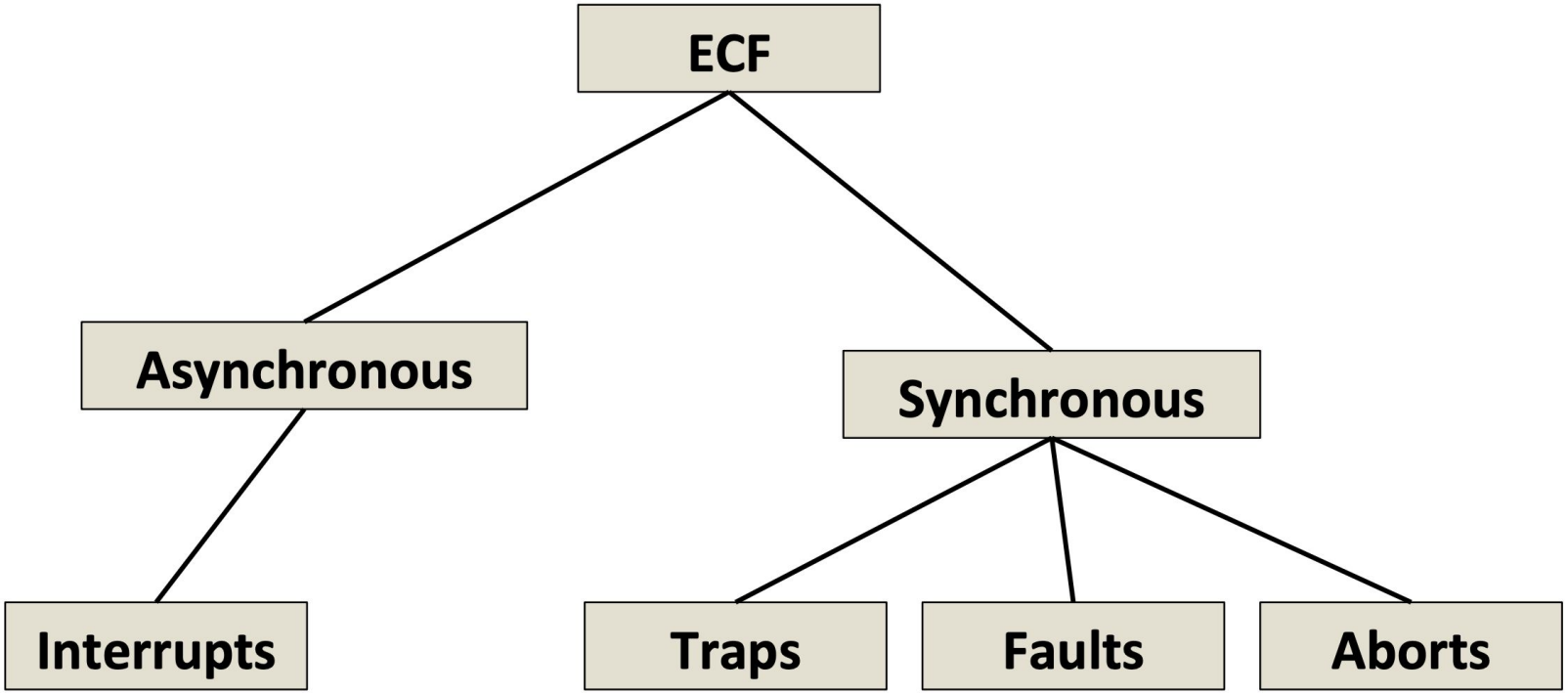
```
lock (lock_a);
lock (lock_b);
b += 5;
a += 7;
a += b;
unlock (lock_a);
b += 11;
unlock (lock_b);
```

Exception

An *exception* is a transfer of control to the OS *kernel* in response to some *event* (i.e., change in processor state)

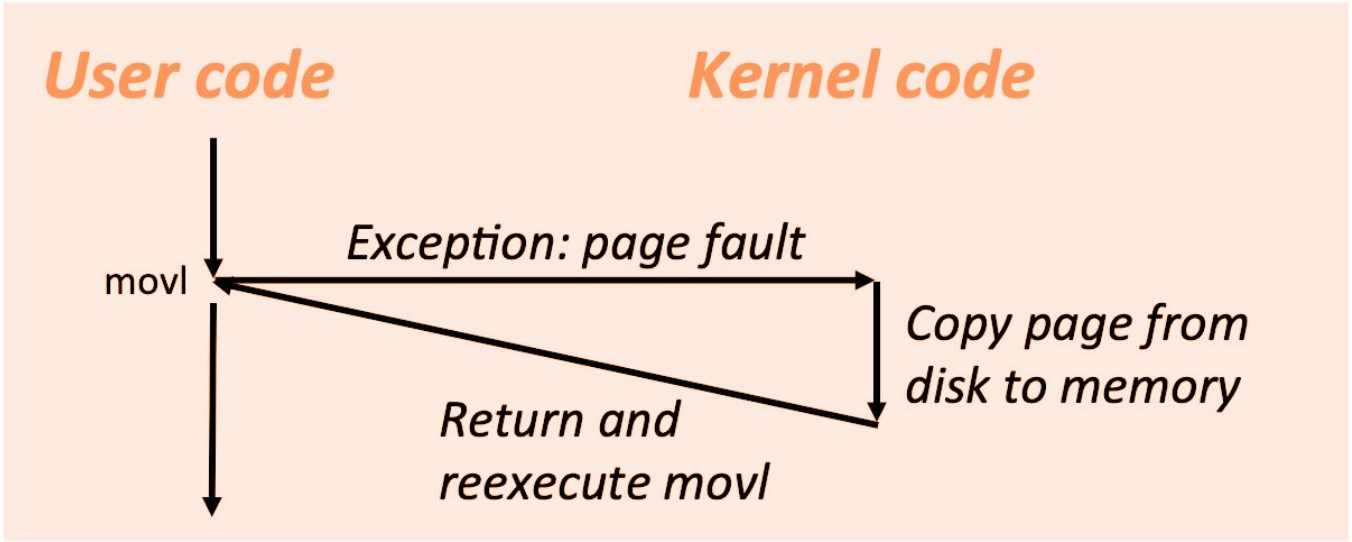


Exception

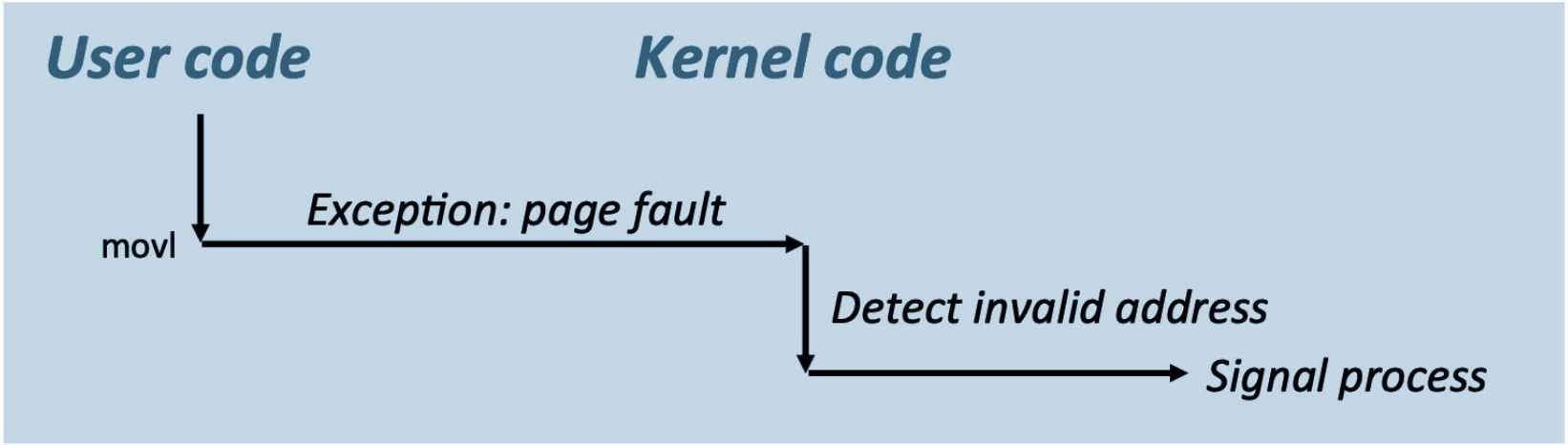

ECF
Asynchronous
Synchronous
Interrupts
Traps
Faults
Aborts

Fault Example: Page Fault

What if there was a page fault?



Fault Example: Invalid Memory Reference



They can also lead to ***process termination***

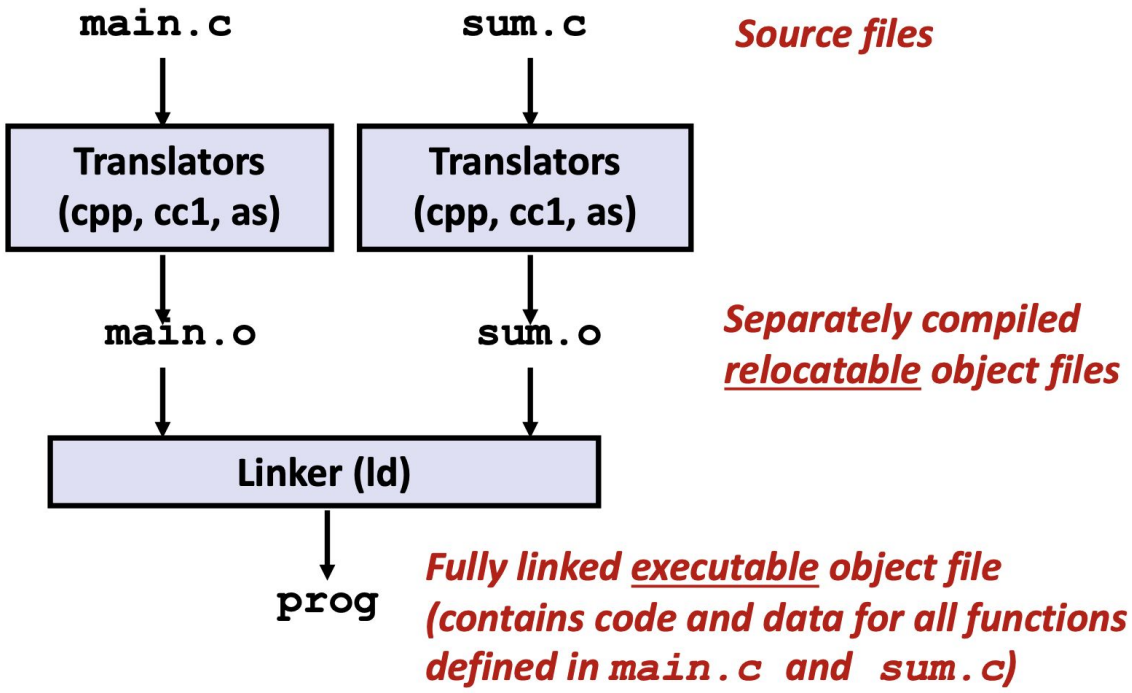
Linking

```
int sum(int *a, int n);

int array[2] = {1, 2};

int main(int argc, char** argv)
{
    int val = sum(array, 2);
    return val;
}
```

main.c

How do we find the **sum** function?

Linker



Why do we use **Linking**?

Linking

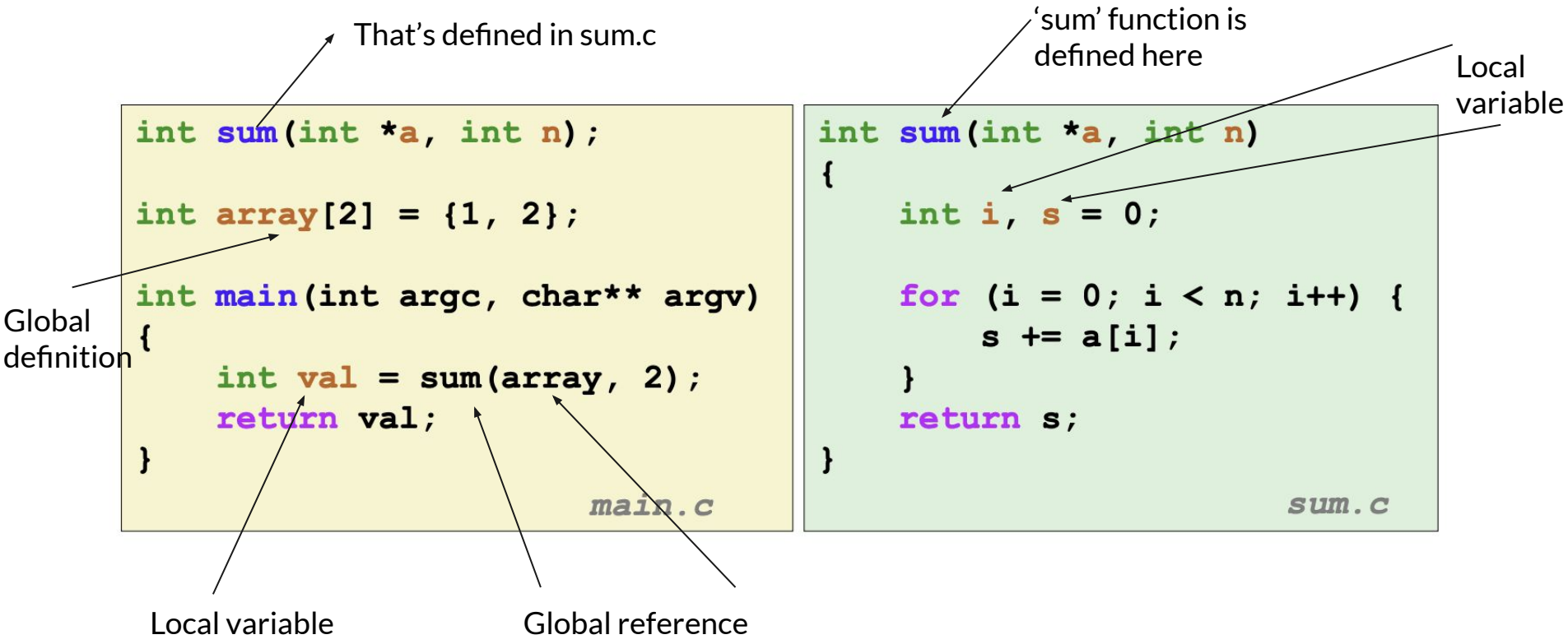
gcc -o prog main.c sum.c

gcc -o prog sum.c main.c

What do linkers do?

- Symbol resolution
- Relocation of code and data section

Example - Symbol Resolution



Check your understanding!

```
int time;

int foo(int a) {
  int b = a + 1;
  return b;
}

int main(int argc,
         char* argv[]) {
  printf("%d\n", foo(5));
  return 0;
}
```

Filename: foo.c

What would be the symbols in **foo.o**?

Check your understanding! - Solution

```c
int time;

int foo(int a) {
  int b = a + 1;
  return b;
}

int main(int argc,
         char* argv[]) {
  printf("%d\n", foo(5));
  return 0;
}
```

Filename: foo.c

What would be the symbols in **foo.o**?

- time
- foo
- main
- printf

More Puzzles

```
int x;
p1() {}
```

```
p1() {}
```

Cannot initialize a function p1 twice

```
int x;
p1() {}
```

```
int x;
p2() {}
```

'x' would refer to the same memory location

```
int x;
int y;
p1() {}
```

```
double x;
p2() {}
```

Write to 'x' in p2 file will override value in 'y' in p1 file

```
int x=7;
int y=5;
p1() {}
```

```
double x;
p2() {}
```

Write to 'x' in p2 file will override value in 'y' in p1 file

```
int x=7;
p1() {}
```

```
int x;
p2() {}
```

'x' might be initialized

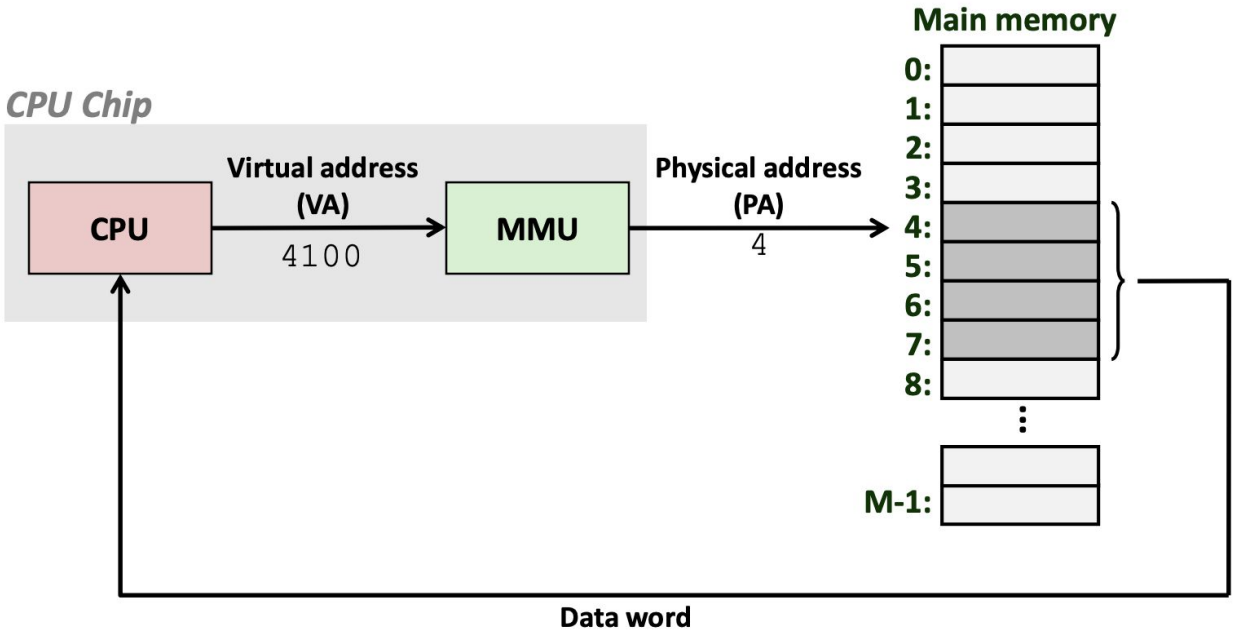
P1

X - mem

Y - mem + 4

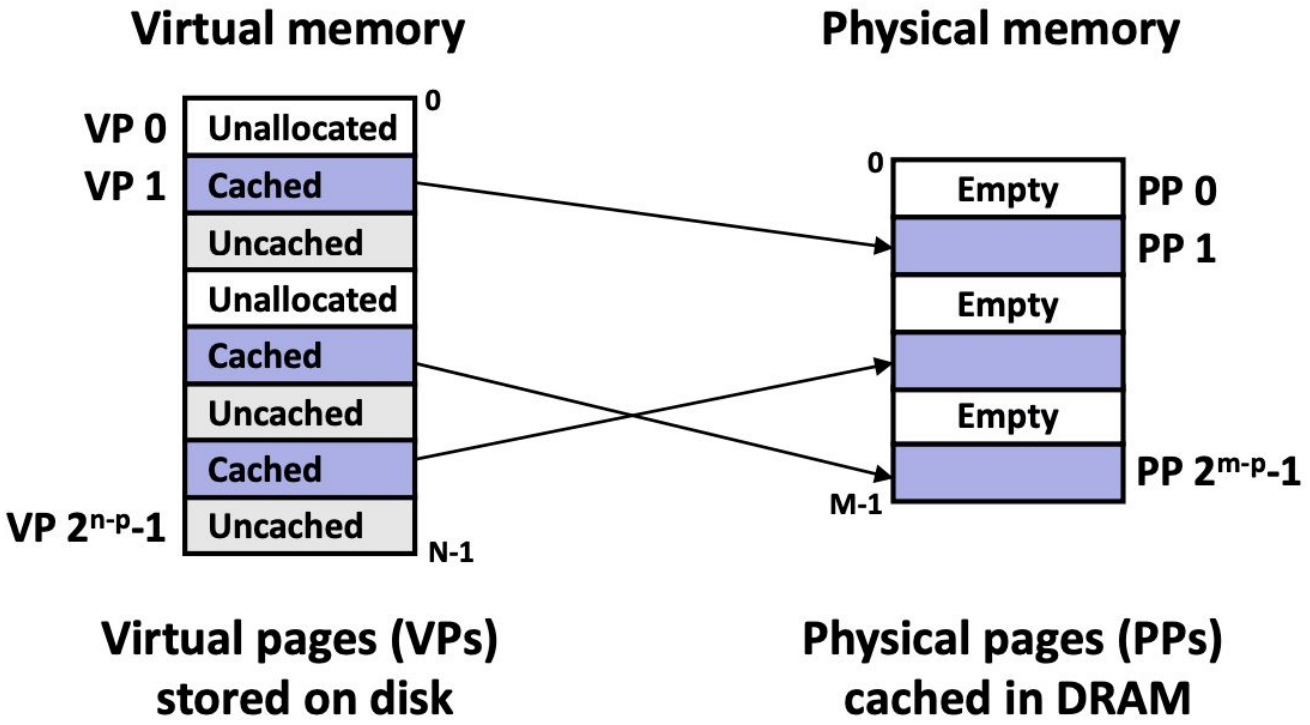
P2: X - mem

Virtual Memory

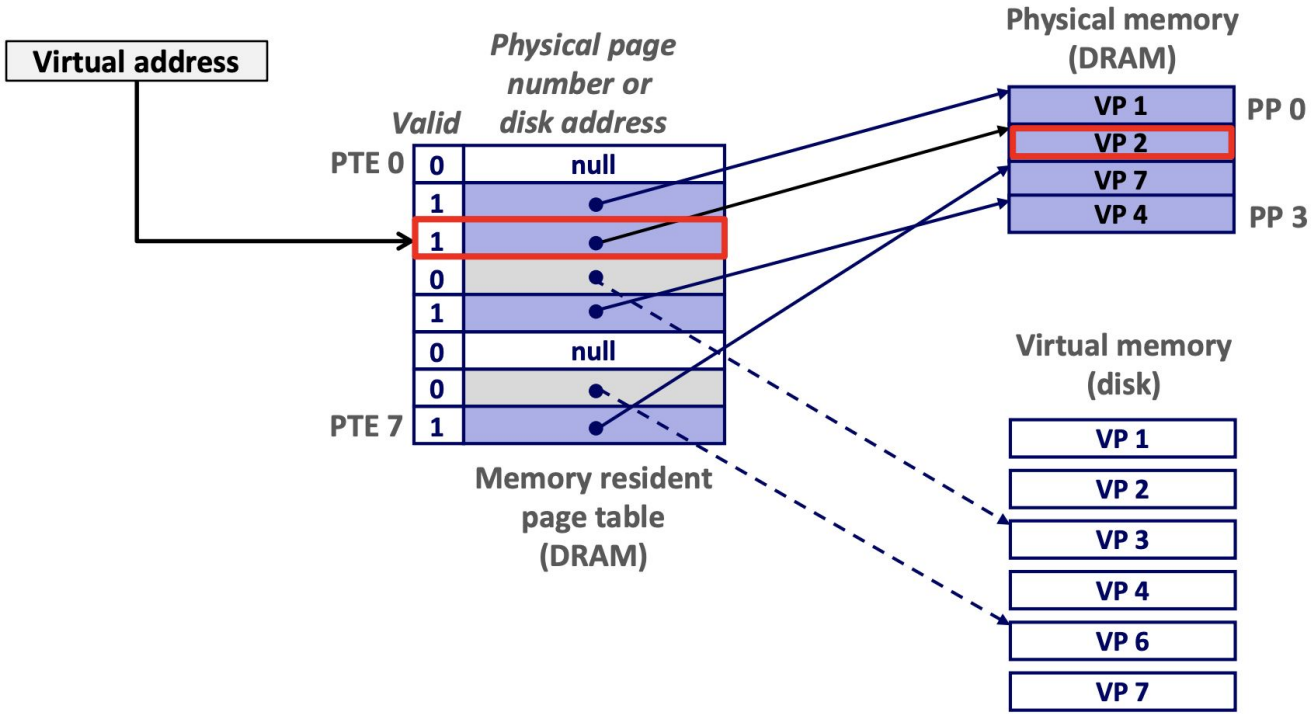


Virtual Memory Addressing

Virtual Memory - Caching



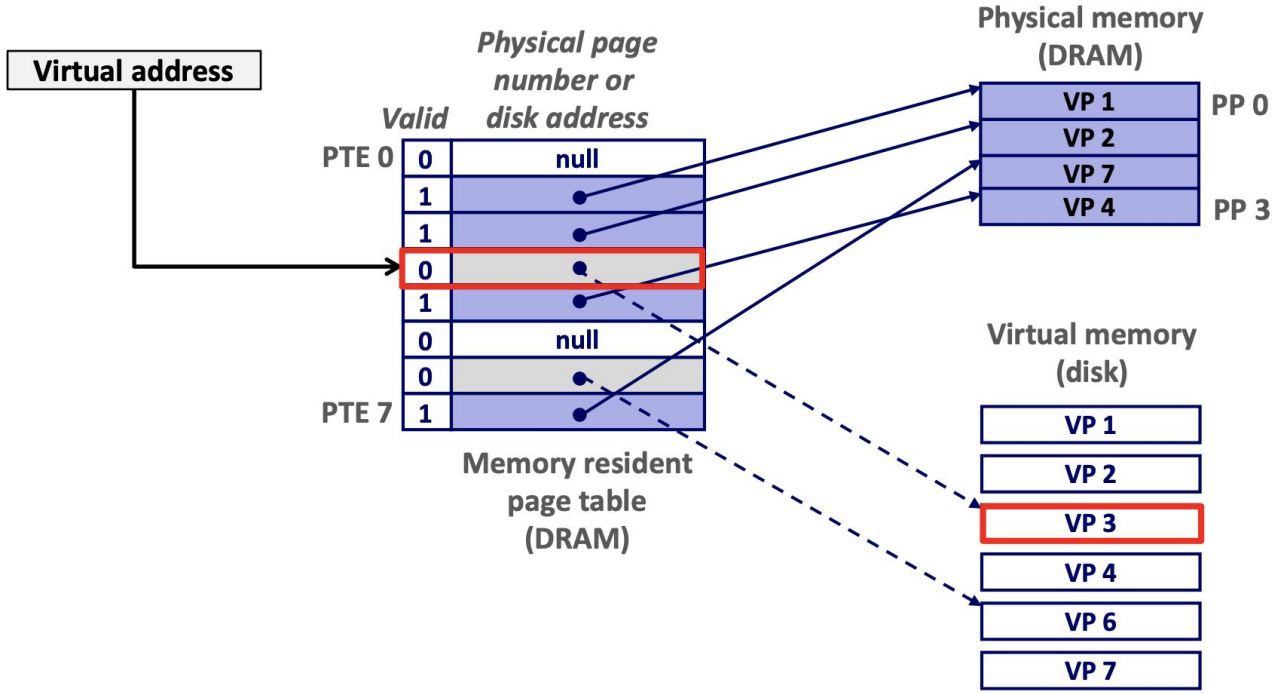
Virtual Memory - Page Hit



We found the address in the main memory!

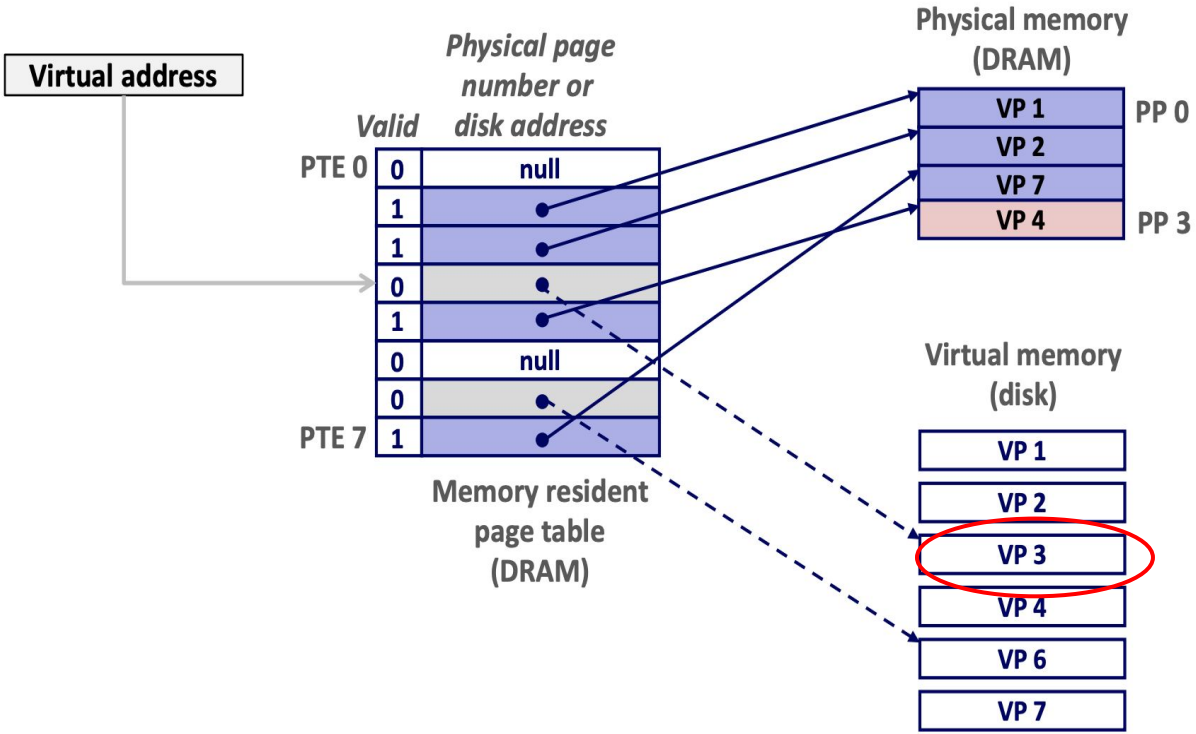
What if we don't find it?

Virtual Memory - Page Fault



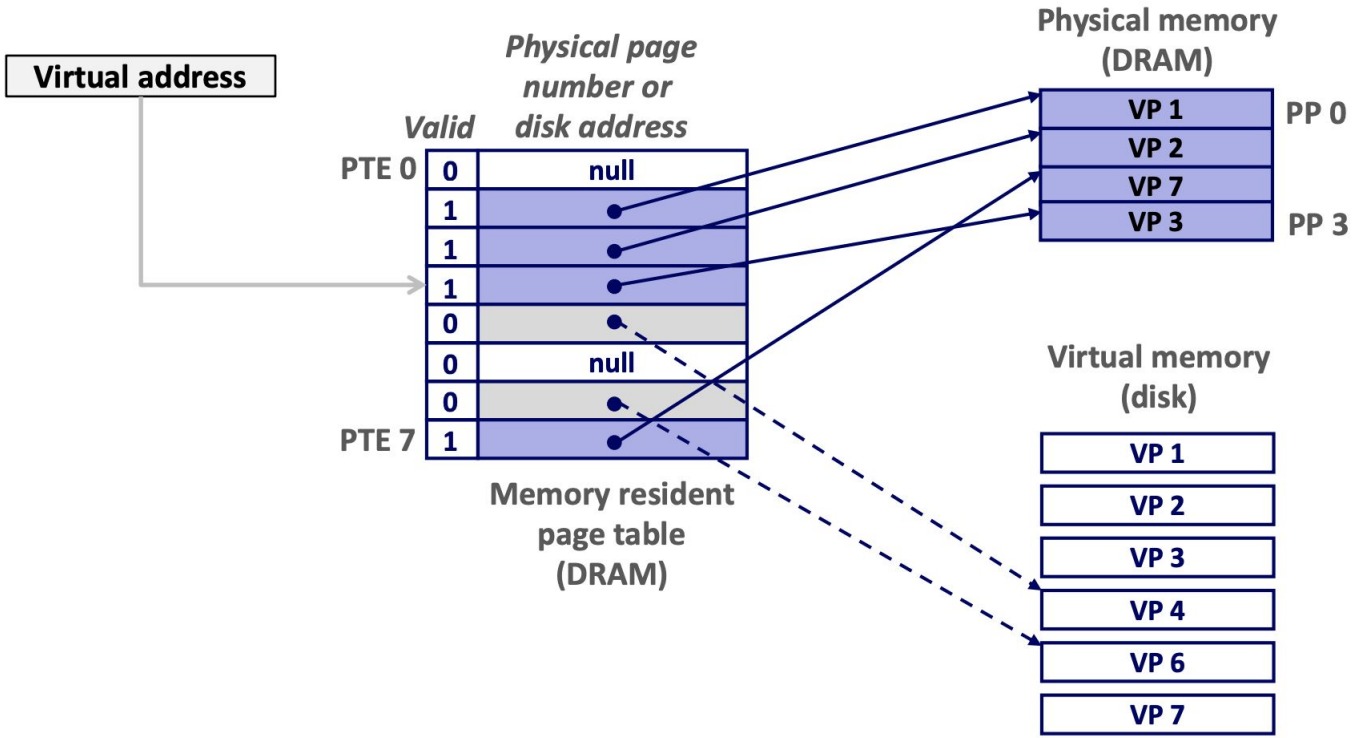
What should we do in this scenario?

Virtual Memory - Handling Page Fault

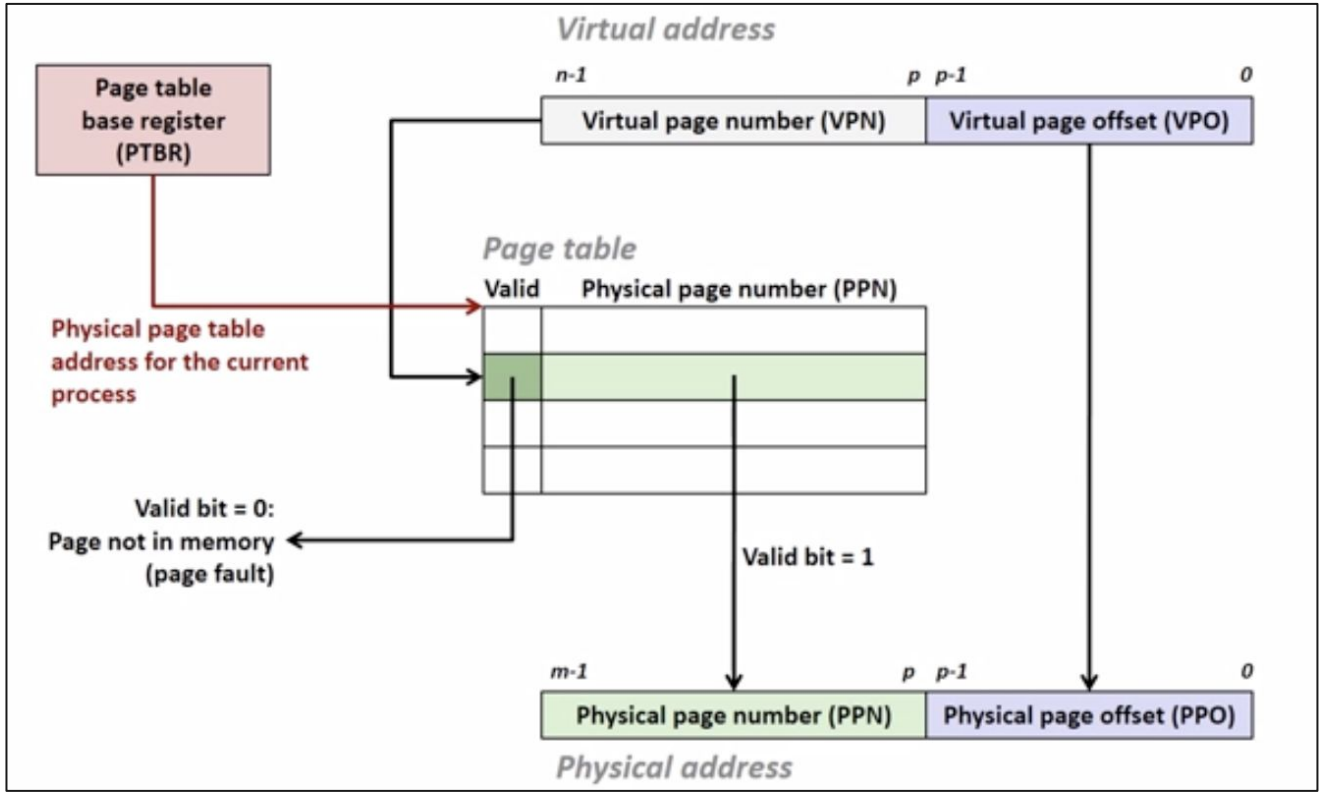


Virtual memory uses the idea of **Demand Paging**

Virtual Memory - Handling Page Fault



Address Translation - Page Table



Check your understanding!

A computer system has a 36-bit virtual address space with a page size of 8K, and 4 bytes per page table entry.

1. How many pages are in the virtual address space?
2. What is the maximum size of addressable physical memory in this system?

Check your understanding! - Solution

A computer system has a 36-bit virtual address space with a page size of 8K, and 4 bytes per page table entry.

1. How many pages are in the virtual address space?

   => 2^36 bytes / (2^3*2^10 page-size) = 2^36/ 2^13 = 2^23 pages

2. What is the maximum size of addressable physical memory in this system?

   => 2^(4*8) = 2^32 pages. Page-size: 2^13 bytes. Size of the physical memory is (2^32*2^13) = 2^45 bytes

Worksheet

https://tinyurl.com/cs33-ucla-bday