Linking

COMP360

“There's a direct ***link*** *between percentage of young people that are educated and how we live our lives.”*

Jeb Bush

Second Exam

- The second COMP360 exam will be in lecture on Wednesday, March 22, 2017

Binding

- A binding is an association between two things, such as a name and the thing it names
- A name is exactly what you think it is
  - Most names are identifiers
  - symbols (like '+') can also be names
- The scope of a binding is the part of the program in which the binding is active

from Programming Language Pragmatics, 4th ed, by Michael Scott

What are we binding?

- At some point a variable name is bound to a memory location
- In Java, the keyword `this` means this object. It is bound to a specific object when the method is executed
- Keywords (such as final, synchronized or long) are bound to a specific concept at language design

Binding Time

- Binding Time is the point at which a binding is created or, more generally, the point at which any implementation decision is made
- Some bindings are made before the program is executed
- Other bindings are made during program execution

from Programming Language Pragmatics, 4th ed, by Michael Scott

Binding Time Examples

- **Language design time** – meaning of keywords
- **Language implementation time** – implementation of types
- **Program writing time** – names to concepts
- **Compile time** – variable names to relative addresses, program statements to machine language
- **Link time** - layout of whole program in memory

The linker is run

A. for all C++ programs
B. for all Java programs
C. for only complex C++ programs
D. none of the above

Address Evolution

Level	Address Type
Source code	Name
Object files	Relative to start of segment
Executable file	Relative to start of executable
During execution	Machine address

Virtual Memory Review

- RAM and programs are divided into fixed sized pages
- The page size is usually fixed for a given architecture, often between 512 -8K bytes
- The pages of a program can be put anywhere in RAM. They do not have to be contiguous.
- The page table keeps track of the physical location of pages
- The page table is indexed by the page number portion of a program address

Virtual Memory

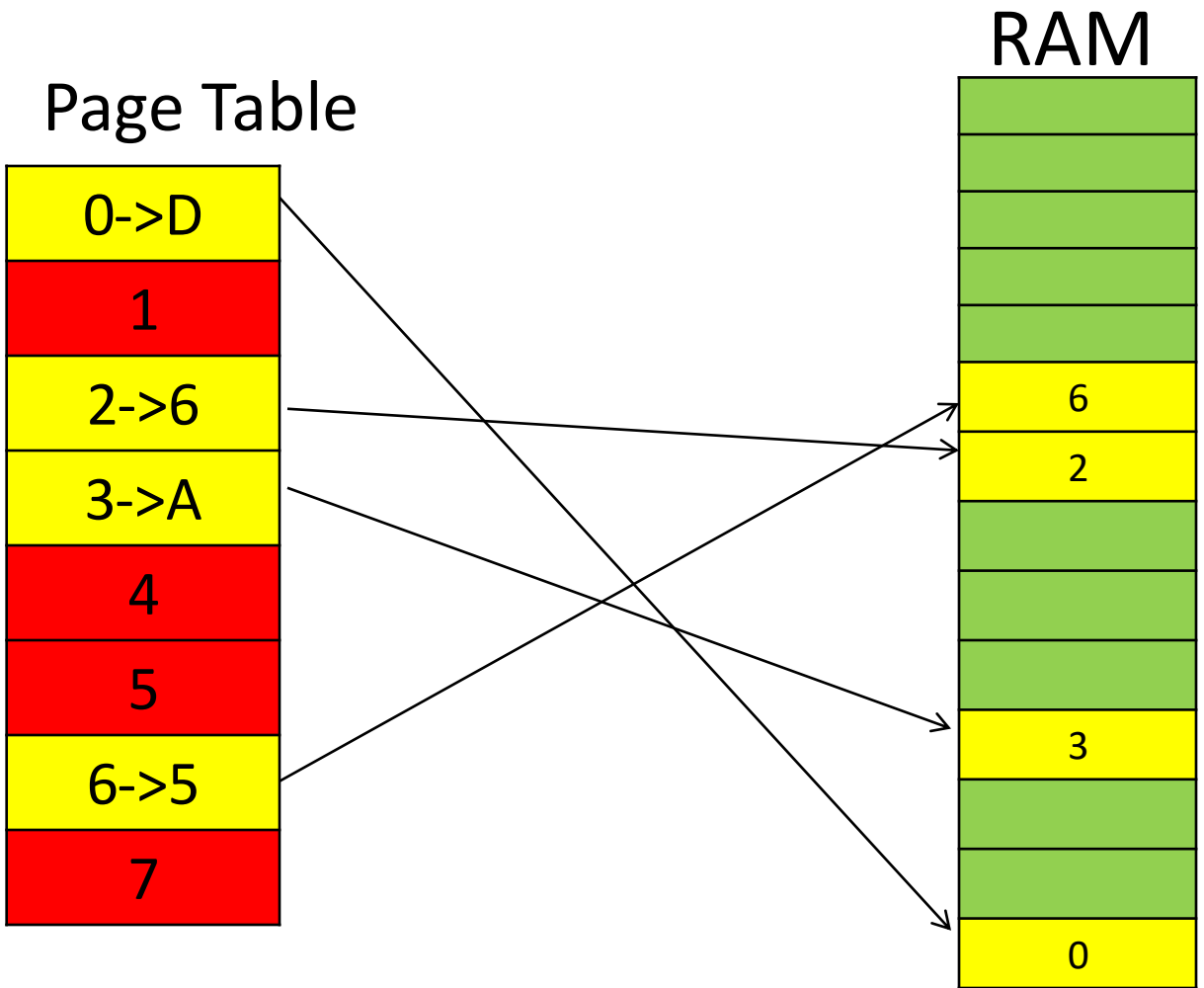
- Virtual Memory is an extension of paging
- Only the pages that are being used are in RAM
- A copy of all pages of a program are in the page file
- If a program accesses an address in a page not in RAM, the hardware creates a page fault interrupt and the OS copies the desired page into RAM

Virtual Memory Implementation

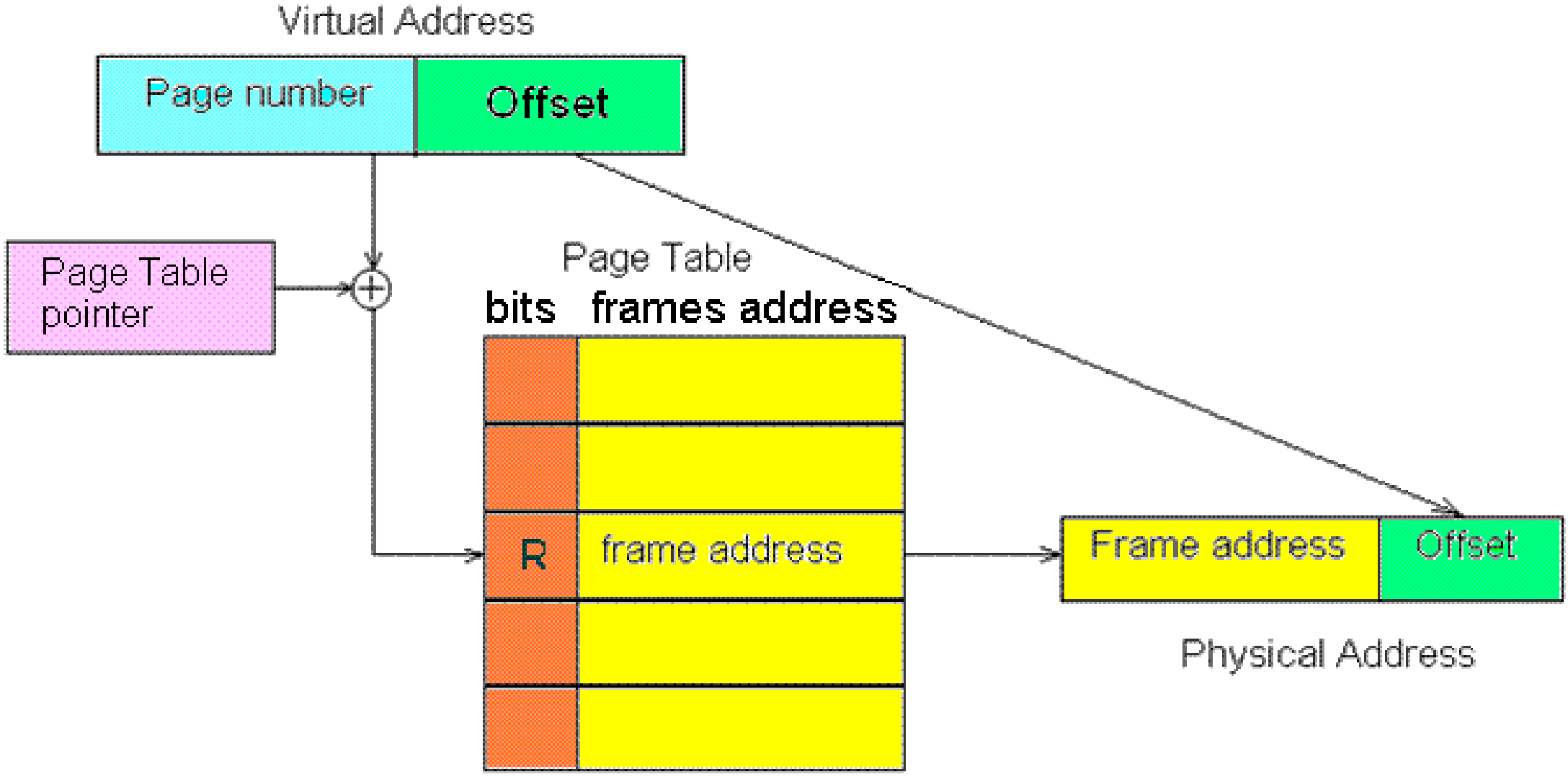
- Unused pages of a program do not need to be in RAM to execute the program
- A “resident” bit is added to the page table
  - Pages in RAM have the resident bit set
  - Pages not in RAM have the resident bit cleared
- All pages are stored on disk
- When a program references a page with the resident bit clear, the hardware creates a page fault interrupt

Only Necessary Pages in RAM

Program addresses showing pages that are referenced



Address Translation



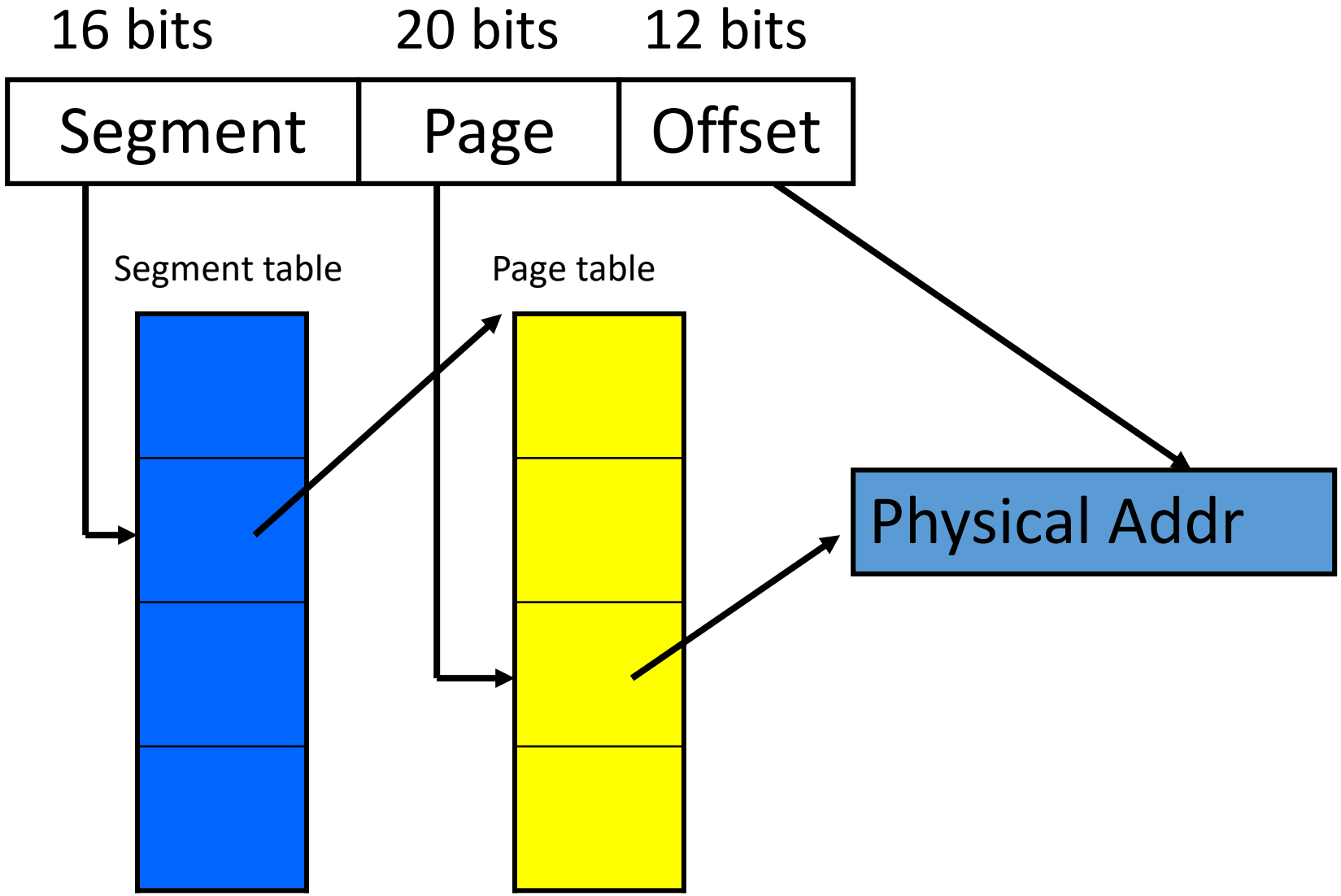
Hardware and Operating System

- Virtual memory requires both hardware and operating system support
- When a page fault interrupt occurs, the OS reads the desired page from disk into an available page of RAM
- The user's page table is updated to point to the newly loaded page
- The program is placed back on the ready list to be executed

Page Table Flags

- Each page table entry has flag bits
  - **Resident** – set if the virtual address is in RAM and clear otherwise
  - **Used** – set when the page is referenced
  - **Dirty** – set when the page is changed
  - **No Execute** – Instructions cannot be fetched from this page
- The Used and Dirty bits are set by the hardware and cleared by the OS

Intel 64bit Segment Addresses

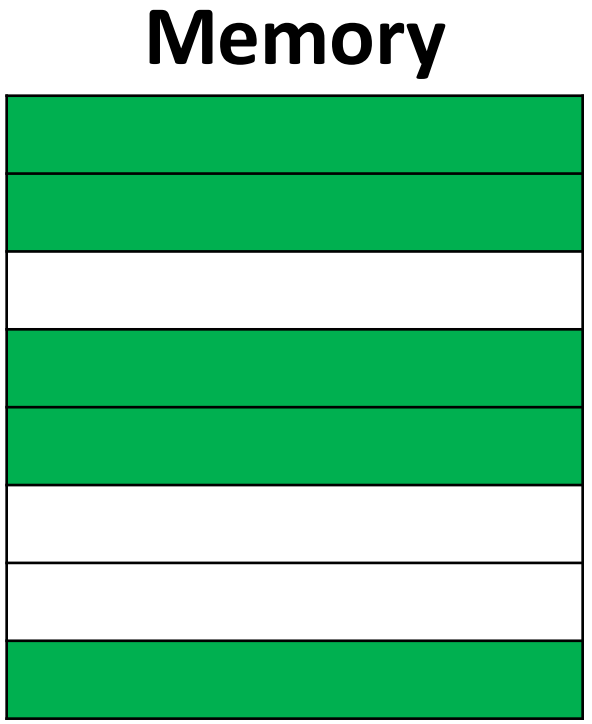


User Programs and Virtual Memory

- Virtual memory is transparent to user programs
- Programs are unaware of page faults
- The major impact of virtual memory on user programs is performance
- If a program creates frequent page faults, the program will run very slowly

Discontiguous Memory

- Not every slot in a page table needs to have a valid entry
- A program may have “holes” in it's address space where no portion of the program is loaded
- Memory segments do not have to be adjacent to save memory



When a program loads an address into a register, when type of address will it contain?

A. Symbolic name
B. Relative to start of segment
C. Relative to start of executable
D. Hardware machine address
E None of the above

Linker

- A **linker** or **link editor** is a program that takes one or more object files generated by a compiler and combines them into a single executable file
- Usually executed by an IDE after a successful compilation
- Input includes:
  - Object files to be included
  - Libraries to be searched for undefined names

Definition from Wikipedia

Compiler Output

```
int             cat  = 47,
extern int dog = 25;
int  myProg(int stuff) {
    int cow = stuff * cat + dog;
```

Addr	Machine Language	Assembler
001e	8b 45 08	mov eax, stuff[ebp]
0021	0f af 05 00 00 00 00 + data seg	imul eax, cat
0028	03 05 00 00 00 00 + dog	add eax, dog
002e	89 45 f8	mov cow[ebp], eax

Object Files

An object file created by a compiler contains:

- Machine language (instruction segment)
  - Addresses require addition of segment base address
- Global data values (data segment)
- List of external names
  - Names that are available to other modules or files
- List of undefined names
  - Names (usually methods) used by not defined

Libraries

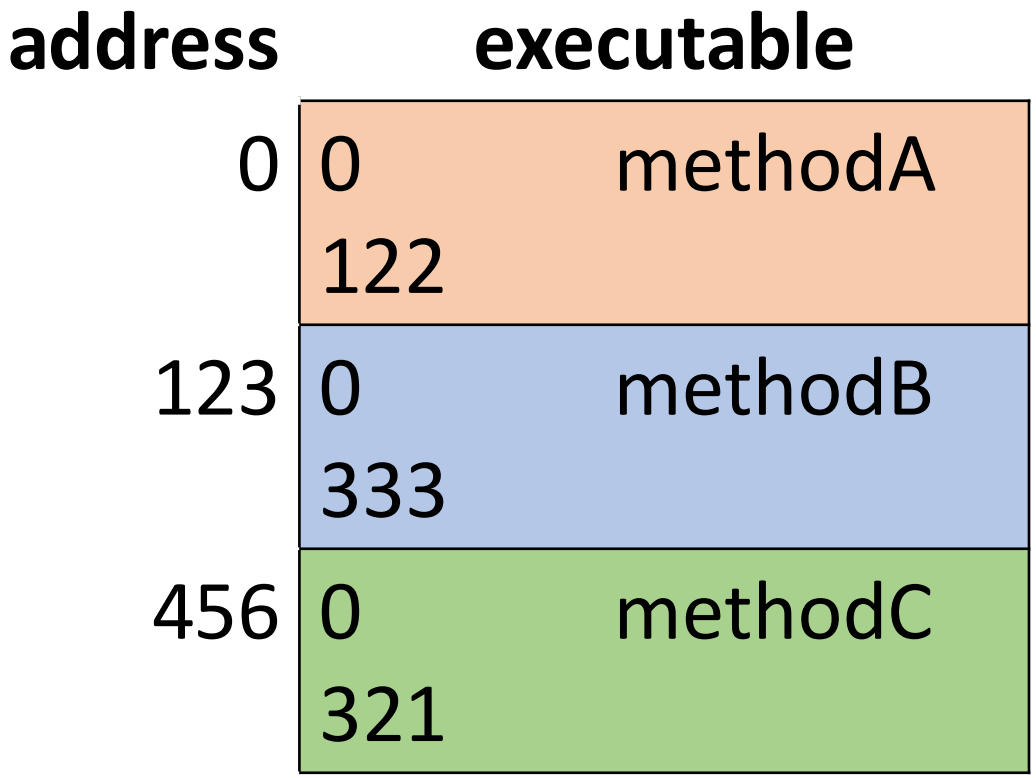
- Libraries are collections of object files
- All of the external names of the object files in the library are indexed to make it easy to search for a name
- The user provides the linker with a list of libraries to search
- Some libraries are implicitly included in the search

Name Resolution

- The linker combines the list of external names and undefined names for all object files
- An external name in one object file may fulfill an undefined name in another object file
- After resolving all undefined names in the included object files, the linker searches the provided libraries for object files that have external names filling the undefined names

Combining Object Files

- The instruction segments of the object files are concatenated into one segment
- This establishes the location and address of the object files in the executable



Adjusting Address in the Code

- After the address of each object file is established in the executable, the addresses in the instructions are adjusted by adding the start address of the appropriate object file
- The address of external names are inserted into the instructions

Other Segments

- The data segments of all object files are combined in a similar manner
- The stack and heap segments do not have any initial values and are allocated at load time

Executable File

- After all names have been resolved and addresses adjusted, the machine language and data are written to the executable (.exe) file
- The linker may write the table of names and addresses to the executable to assist debuggers

Program Execution

- When an executable is to be run, memory pages are allocated, the page table is built and the segments from the exe file are read into memory
- Uninitialized data is allocated, but nothing is read into these addresses
- Only some of the pages are actually loaded into memory. The others are loaded when they are referenced
- Memory space is allocated for the stack and heap

In the data segment

A. Name resolution is identical to the instruction segment
B. There is no name resolution in the data segment
C. There are only external names, not undefined
D. All the above
E None of the above

Linking into Memory

- Some systems directly load the object files into memory instead of writing an executable file
- The linking process is the same, but the result is not saved

Shared Libraries

- Some functions are commonly used by many programs
- To avoid having many identical copies of these methods in memory, most systems provide a mechanism to share the instruction segment
- The function is loaded into memory upon first use
- When other programs want to use the function, an entry in the program’s page table is created to point to the shared memory segment

Runtime Loading

- Some compiled programs are not linked until execution
- In the Java Virtual Machine, a class is not loaded until it is needed
- It is possible that a class does not exist when the program is executed, thus generating an error

Loading Many Classes for any Java Program

- Java loads 415 classes to run

```
public class Minimal {
   public static void main(String[] unused) {
      return;
   }
}
```

ClassLoader

- Java programs can extend the java.lang.ClassLoader class
- Class loaders can read the requested class from files, the network or can generate the class dynamically
- Users might use their own class loader for security, debugging, profiling or dynamic generation

Resolving Names or Signatures

- For traditional languages, the linker would match the name of a called function to the functions address
- In object oriented languages, the linker matches the function or method's full signature including method name, return type and parameter types
- This supports polymorphism:

```
int myMethod( int cat )
```

is different from

```
int myMethod( String cat )
```

Second Exam

- The second COMP360 exam will be in lecture on Wednesday, March 22, 2017