NEW YORK UNIVERSITY

CSCI-GA.2130-001

Compiler Construction

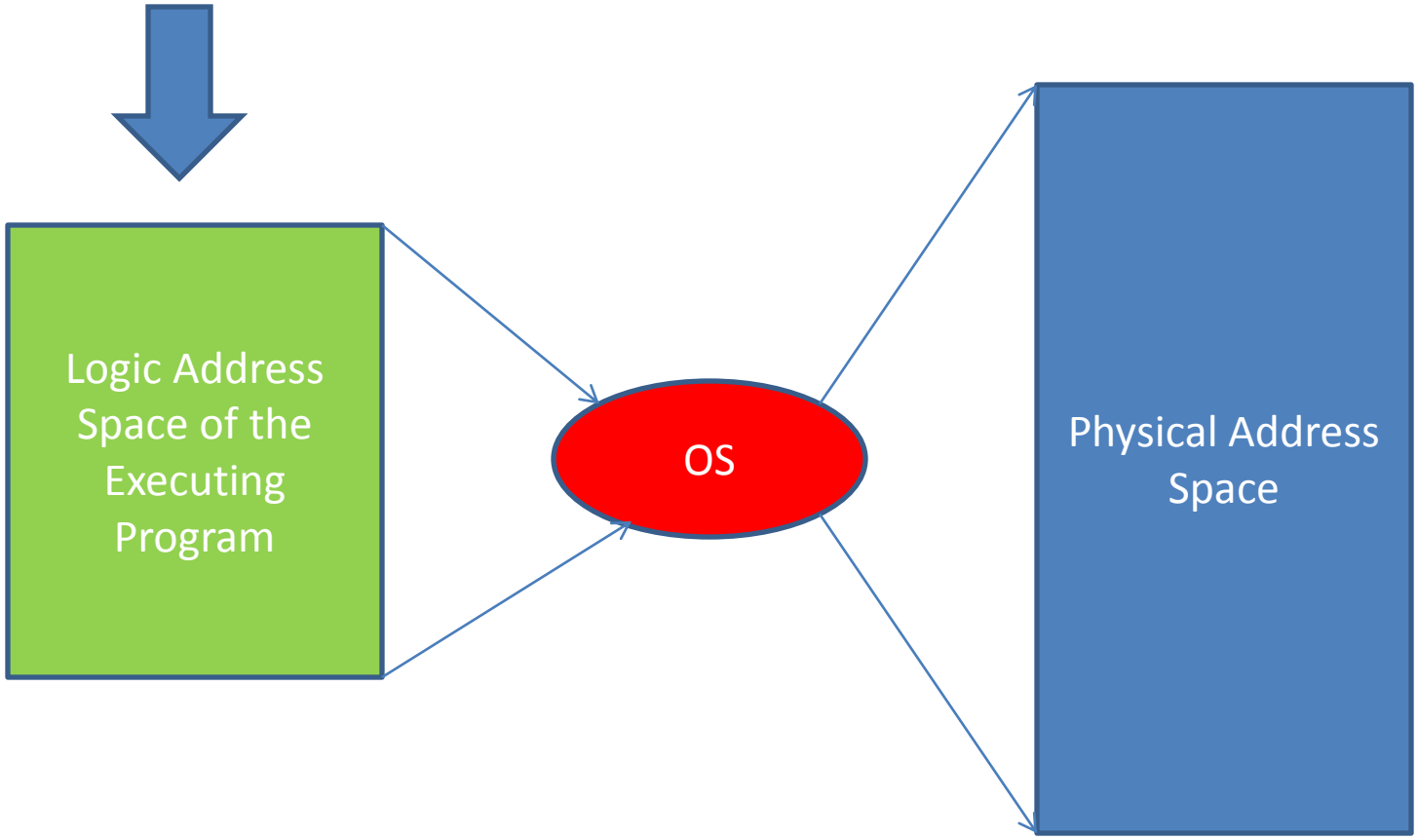
Lecture 11: Run-Time Environment

Mohamed Zahran (aka Z)
mzahran@cs.nyu.edu

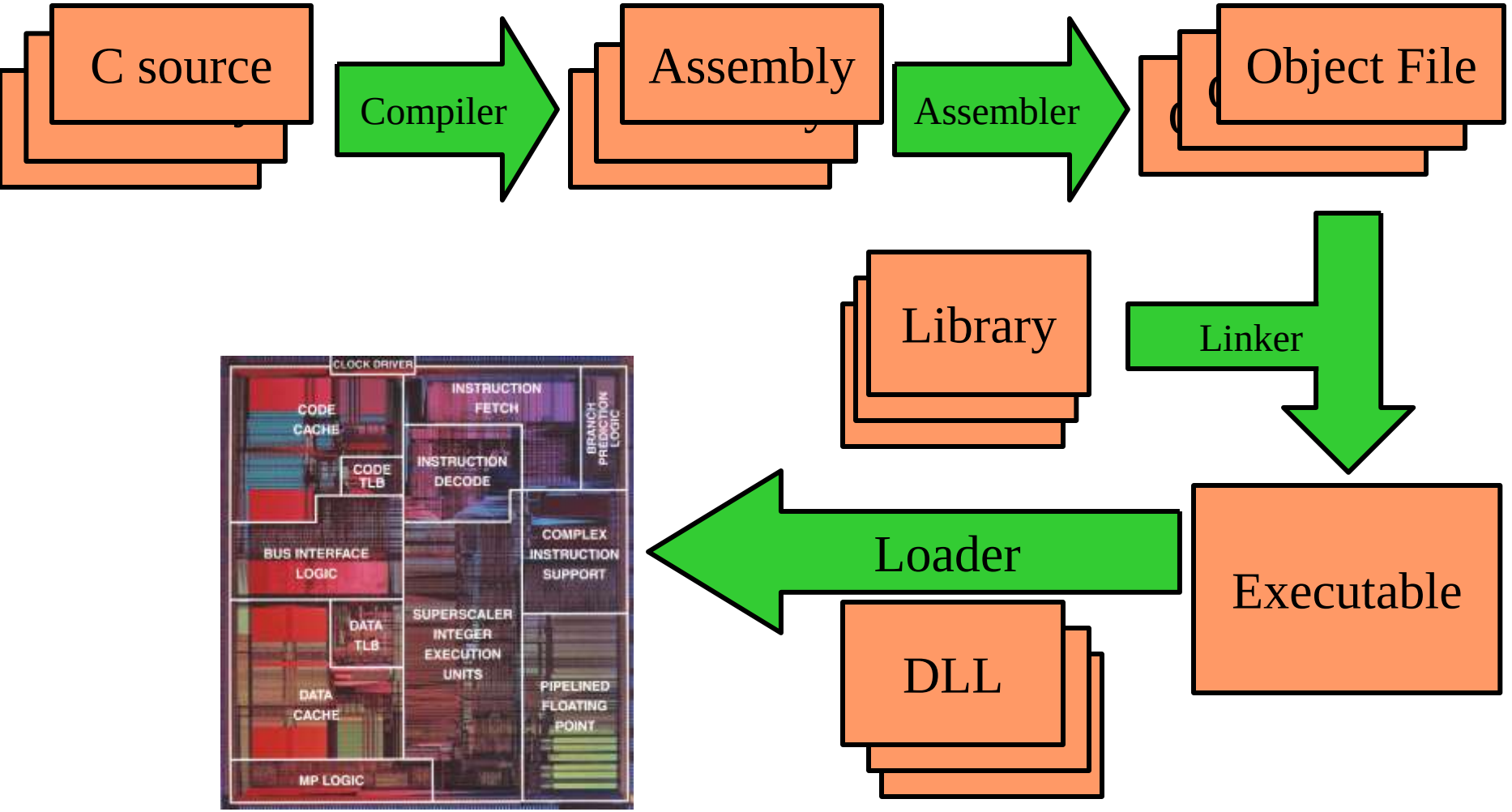
What Are We Talking About Here?

- How do your code and data look like during execution?
- Interaction among compiler, OS, and target machine
- The main two themes:
  - Allocation of storage locations
  - Access to variables and data

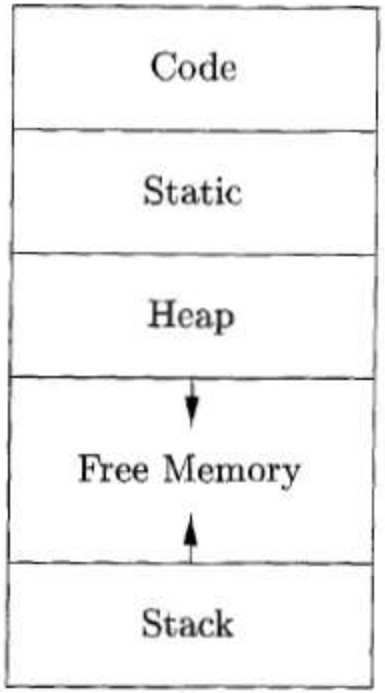
Compiler-writer Perspective



Source Code to Execution



Typical Memory Subdivision



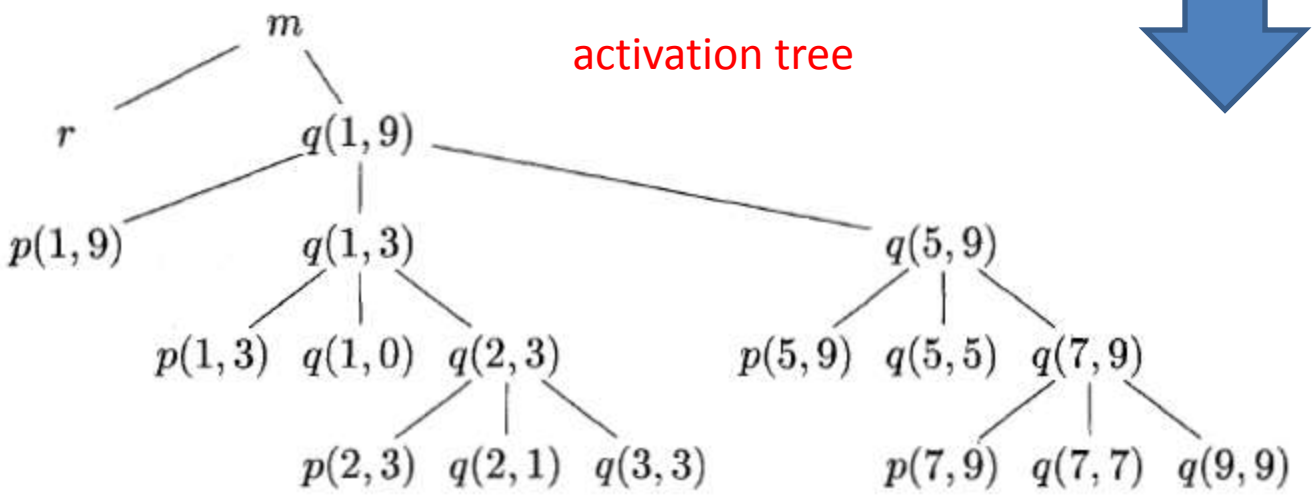
Stack Allocation

- For managing procedure calls
- Stack grows with each call and shrinks with each procedure return/terminate
- Each procedure call *pushes* an activation record into the stack

```
int a[11];
void readArray() { /* Reads 9 integers into a[1], ..., a[9]. */
    int i;
    ...
}
int partition(int m, int n) {
    /* Picks a separator value v, and partitions a[m..n] so that
       a[m..p-1] are less than v, a[p] = v, and a[p+1..n] are
       equal to or greater than v. Returns p. */
    ...
}
void quicksort(int m, int n) {
    int i;
    if (n > m) {
        i = partition(m, n);
        quicksort(m, i-1);
        quicksort(i+1, n);
    }
}
main() {
    readArray();
    a[0] = -9999;
    a[10] = 9999;
    quicksort
}
```

```
enter main()
    enter readArray()
    leave readArray()
    enter quicksort(1,9)
        enter partition(1,9)
        leave partition(1,9)
        enter quicksort(1,3)
            ...
        leave quicksort(1,3)
        enter quicksort(5,9)
            ...
        leave quicksort(5,9)
    leave quicksort(1,9)
leave main()
```

activation tree



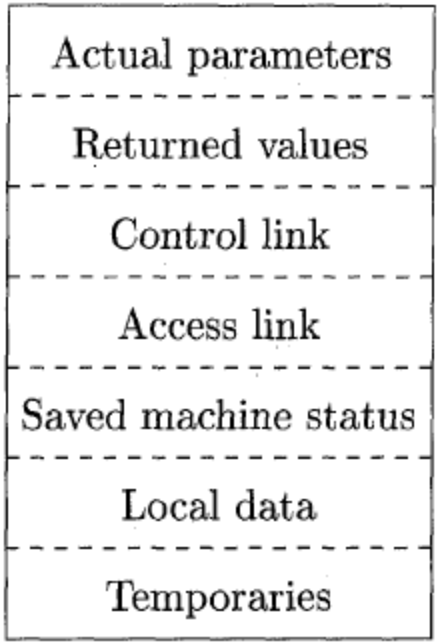
Activation Tree

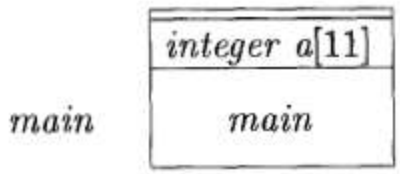
- Models procedure activations
- The *main* is the root
- Children of the same parent are executed in sequence from left to right
- Sequence of procedure calls -> preorder traversal of activation tree
- Sequence of procedure returns -> postorder traversal of activation tree

Activation Records

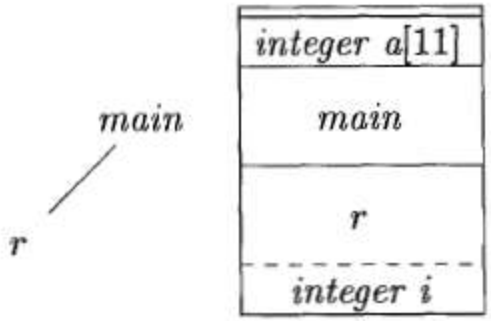
- What is pushed into the stack for each procedure activation
- Contents vary with the language being implemented

General Activation Record

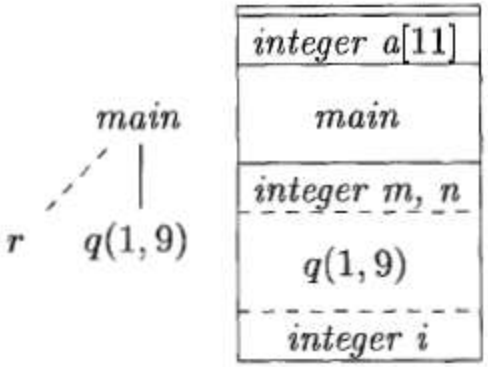




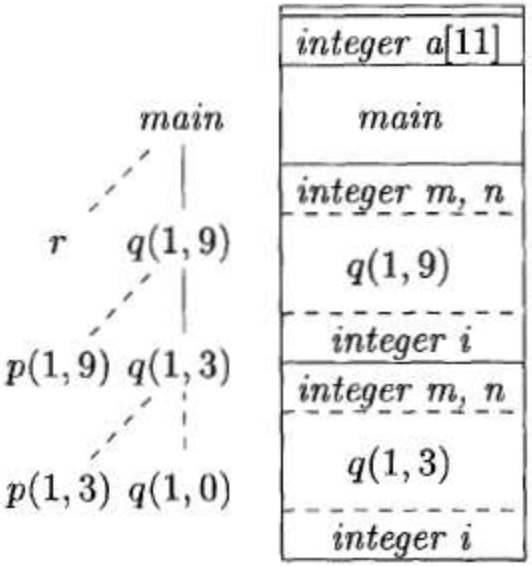
(a) Frame for *main*



(b) r is activated



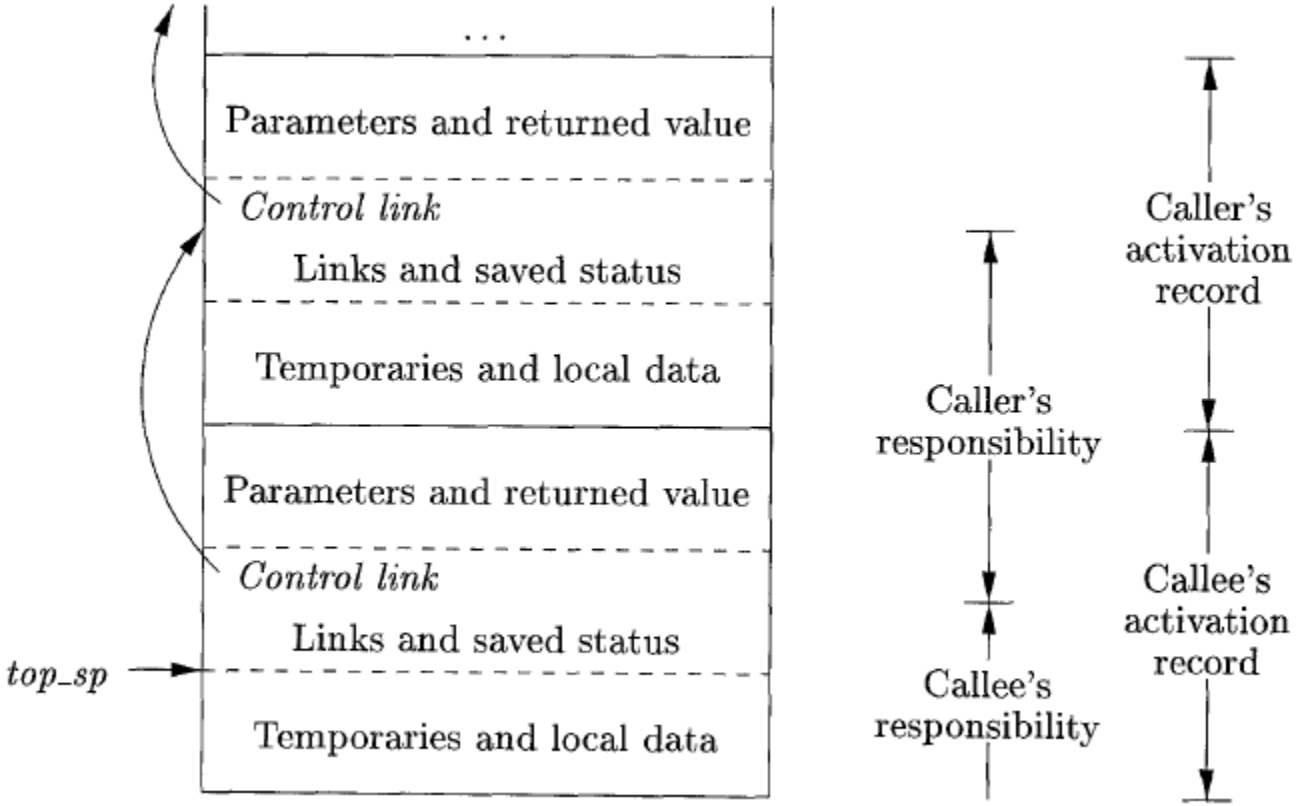
(c) r has been popped and $q(1,9)$ pushed



(d) Control returns to $q(1,3)$

Code Generation

- Calling sequence
  - Code that allocates activation record
  - Code for entering information in it
- Return sequence
  - Code to restore the state of the machine

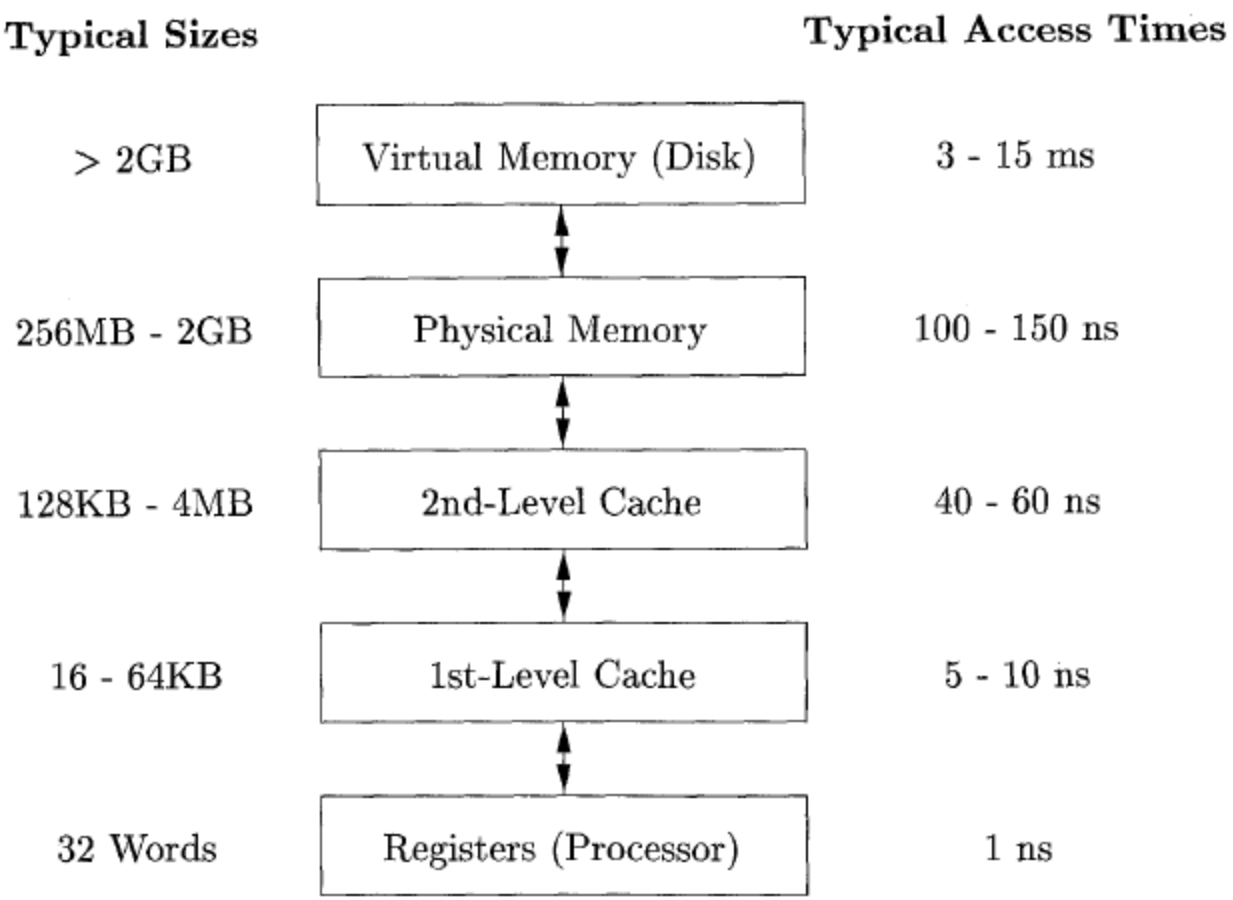

...
Parameters and returned value
Control link
Links and saved status
Temporaries and local data
Parameters and returned value
Control link
Links and saved status
top_sp
Temporaries and local data
Caller's responsibility
Callee's responsibility
Caller's activation record
Callee's activation record

Heap Management

- Heap: portion of the store used for data that lives indefinitely
- Memory manager: subsystem responsible for (de)allocation of space within the heap
- Garbage collection: process of finding spaces within the heap that are no longer used and reallocate them to other data items

Memory Manager

- Keeps track of all the free space in heap at all time
- Allocation
  - Interaction with OS
- Deallocation
- Desired properties:
  - Space efficiency: minimize total heap space needed by programs
  - Program efficiency: making good use of memory subsystem
  - Low overhead: of (de)allocation processes


Typical Sizes
Typical Access Times
> 2GB
Virtual Memory (Disk)
3 - 15 ms
256MB - 2GB
Physical Memory
100 - 150 ns
128KB - 4MB
2nd-Level Cache
40 - 60 ns
16 - 64KB
1st-Level Cache
5 - 10 ns
32 Words
Registers (Processor)
1 ns

Heap Fragmentation

- Due to allocation/delallocation
- Why is it bad?
- How to deal with it?
  - Best fit
  - First fit
  - Next fit
  - Worst fit

Garbage Collection

- Garbage: data that cannot be referenced
- Garbage collection: reclamation of garbage from heap

Assumptions

- Objects have a type that can be determined by garbage collector at run-time.
- References to objects are always to the address of the beginning of the object.

Performance Metrics

- Overall execution time: garbage collection can be very slow
- Space usage: must avoid fragmentation
- Maximum pause time must be minimized
- Program locality

Reference-Counting Garbage Collection

- Every object must have a field for reference count
- This field counts the number of references to the object
- If count reaches zero, the object is deleted

So

- Skim: 7.3, 7.5.2, 7.6, 7.7, and 7.8
- Read: the rest of chp 7