

USMA SIGSAC

```
12     return 0;
13 }
14 int fib_seq(int s1, int s2, int n) {
15     int sequence[n];
16     sequence[0] = s1;
17     sequence[1] = s2;
18     for(int i = 2; i < n; i++) {
19         sequence[i] = sequence[i-1] + sequence[i-2];
20     }
21     return sequence[n-1];
22 }
23 int shmoocon(int one, int two, int three) {
24     int returnVar;
25     if(one==1) {
26         returnVar = two+three;
27     } else if(one==2) {
28         returnVar = two*three;
29     } else if(one==3) {
30         returnVar = fib_seq(two, three, 10);
31     } else returnVar = 0;
32     return returnVar;
33 }
34
```

Assembly code for function fib_seq (lines 12-22):

```
12     mov     eax, [esp+0x10]
13     mov     DWORD PTR [esp+eax], 0
14     call    0x804835c <gets@plt>
15     mov     eax, 0x8048585
16     lea     edx, [esp+0x10]
17     mov     DWORD PTR [esp+0x4], edx
18     mov     DWORD PTR [esp], eax
19     call    0x804838c <printf@plt>
20     mov     eax, 0x0
21     leave
22     ret     0, 0, 0, 0
```

Assembly code for function shmoocon (lines 23-34):

```
23     jmp     DWORD PTR ds:0x80496a0
24     push    0x20a, 0x20b, 0x20c
25     jmp     0x804833c
26     mov     eax, 0x0
27     call    0x804835c
28     jmp     DWORD PTR ds:0x8049694
29     push    0x8, 0x9, 0xa
30     jmp     0x804833c
31     jmp     DWORD PTR ds:0x80496a0
32     call    0x804838c
33     xchg    edx, eax
34     add     DWORD PTR [eax+ecx*1], 0x0
```

SIGSAC

- Semester Plans:
 - Capture the Flag (CTFs)
 - Cyber Fast Track Talks
 - Haxathon
 - Lessons: Assembly, SQL, Lock picking, back track
 - Personal Projects (and IRC)

SIGSAC – “Safety Brief”



<http://www.panicmanual.com/wp-content/uploads/2010/12/Panic-Manual-Spider-man-teaches.jpg>

SIGSAC – “Safety Brief”

1. Anything you learn here is not to be used maliciously.
2. No poking around in the USMA/DOD networks.
3. Use VMs (and do not NAT them)
4. USMA's detection is better than you think. so no breaking #2, even if you think you're clever.
5. Being a part of SIGSAC is not an excuse to break rules.

SIGSAC – Website

- <http://usmasigsac.com/>
- On the website you will [soon be able to] find:
 - Schedule
 - IRC
 - Haxathon portal
 - Resources
 - Hidden Easter Eggs (added regularly)

SIGSAC – The computer

■ What are the basic components?

```
return 0;
}
int sequence[n/2];
sequence[0] = s1;
sequence[1] = s2;
for(int i = 2; i < n; i++) {
    sequence[i] = sequence[i-1] + sequence[i-2];
}
return sequence[n-1];
}
int shmoocon(int one, int two, int three) {
    int returnVar;
    if(one==1) {
        returnVar = two+three;
    } else if(one==2) {
        returnVar = two*three;
    } else if(one==3) {
        returnVar = fib_seq(two, three);
    } else returnVar = 0;
    return returnVar;
}
```


SIGSAC – The computer

- What are the basic components?

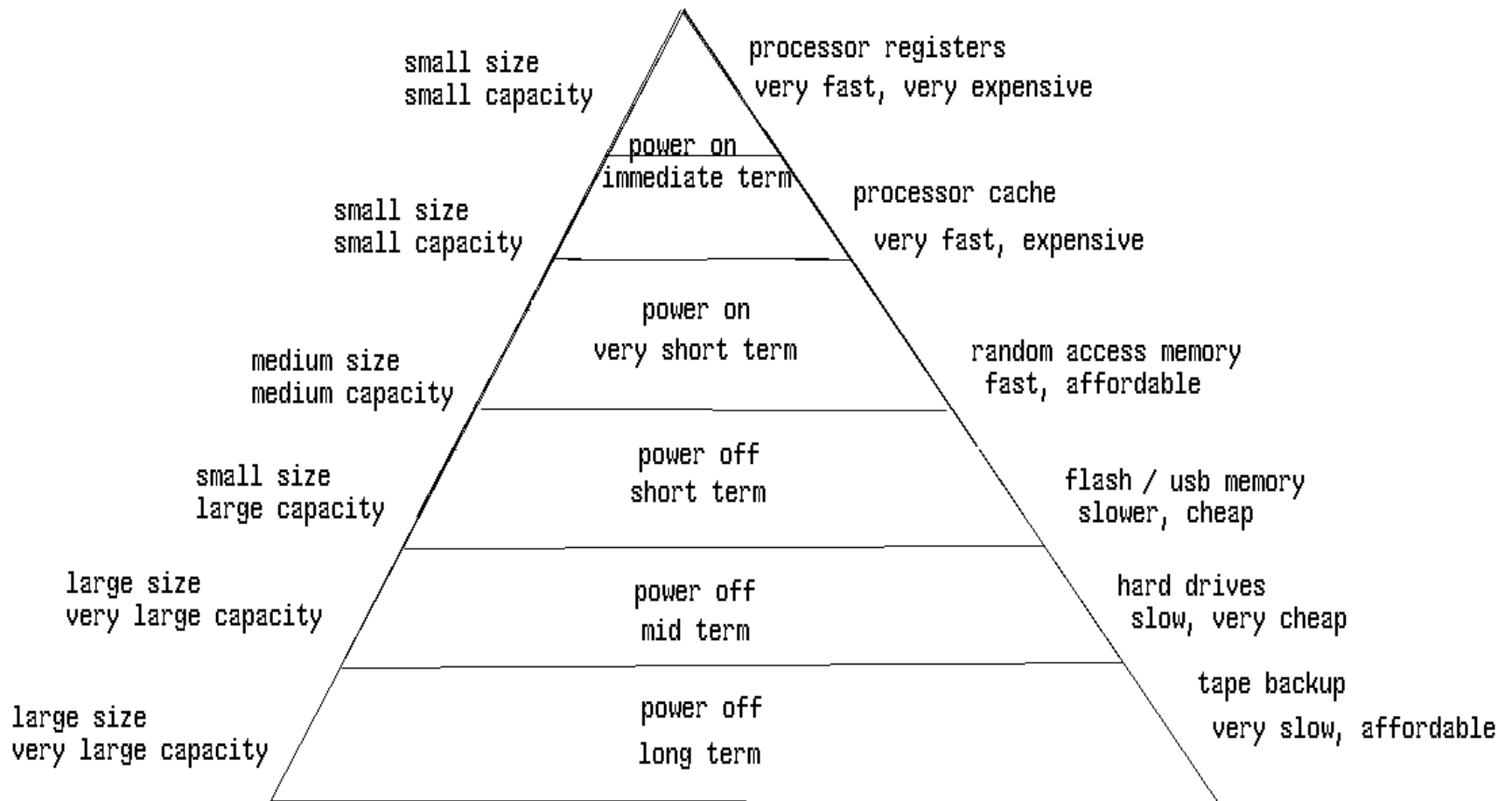
- CPU

- Memory

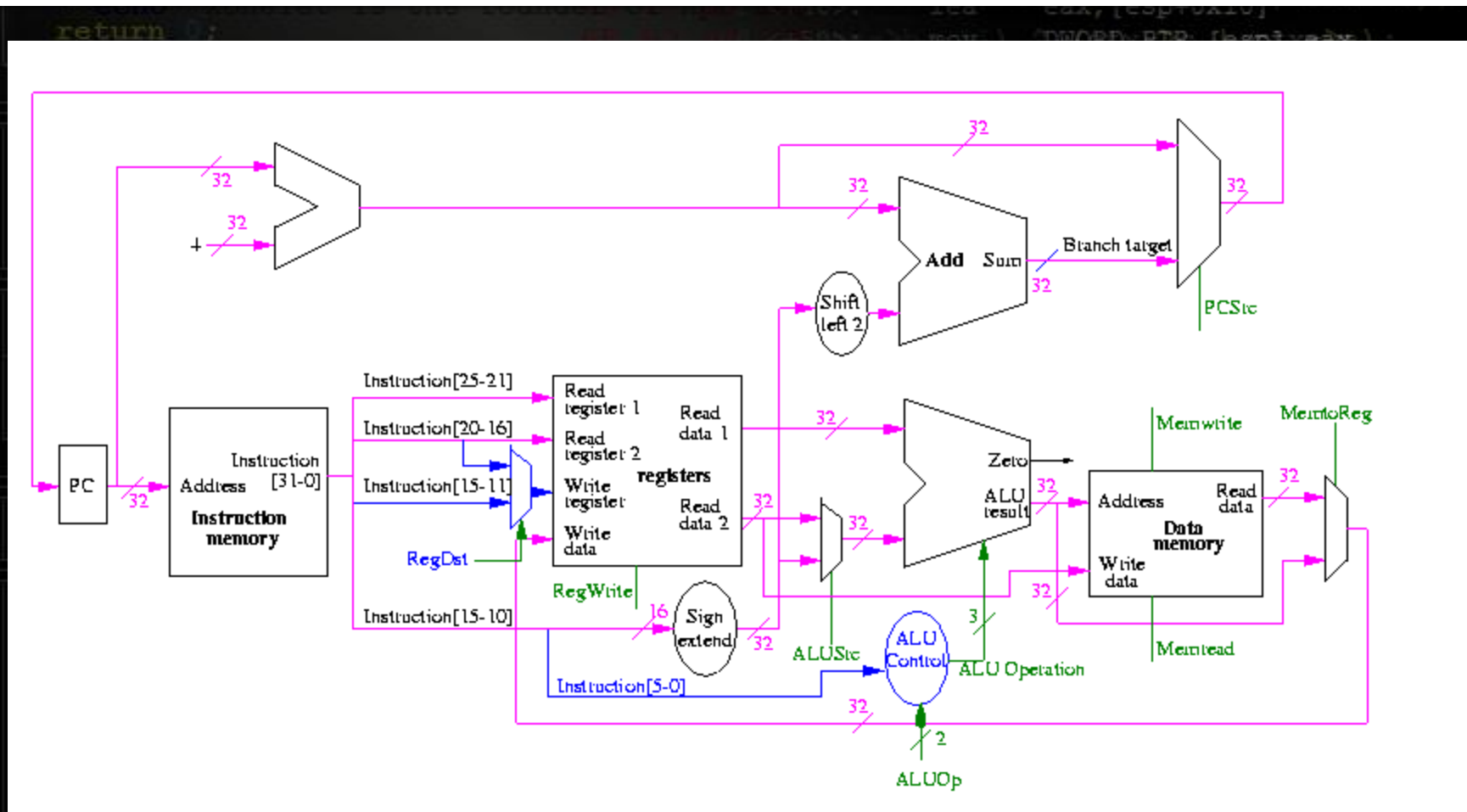
■ Secondary Storage

Computer Memory Hierarchy

by Dan Lash (.com)



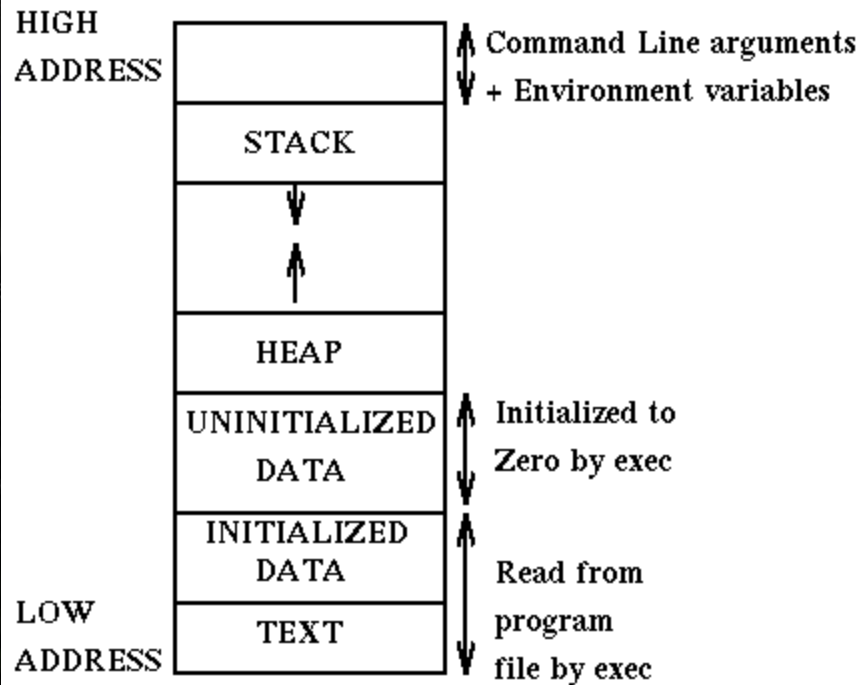
SIGSAC – The computer



<http://cs.nyu.edu/~gottlieb/courses/1999-00-fall/arch/datapath-final-small.png>

SIGSAC – The computer

■ Memory



MEMORY ORGANIZATION PER PROCESS

http://www.cs.rit.edu/~hpb/Lectures/SIA/OS1/UsedGif/5_heap_and_stack.gif

SIGSAC – The computer

```
12     return 0;
13 }
14
15 /* memory example. modeled after pg 183 of gray hat hacking. */
16
17 int an_integer = 5; // stored in data as initialized vlaue
18 char * str; // stored in bss as uninitialized
19
20 void function1(int c) {
21     int i = c; // stored in the stack
22
23     str = (char*) malloc (10 * sizeof (char)); // reserving 10 characters in the heap
24     strncpy(str, "test", 4); // copies 4 characters into str
25 }
26
27 void main() {
28     function1(42);
29 }
30
31 returnVar = fib_seq(two, three);
32 } else returnVar = A;
33 return returnVar;
34 }
```


SIGSAC – The computer

Register Category	Register Name	Purpose
General registers	EAX, EBX, ECX, EDX	Used to manipulate data
	AX, BX, CX, DX	16-bit versions of the preceding entry
	AH, BH, CH, DH, AL, BL, CL, DL	8-bit high- and low-order bytes of the previous entry
Segment registers	CS, SS, DS, ES, FS, GS	16-bit, holds the first part of a memory address; holds pointers to code, stack, and extra data segments
Offset registers		Indicates an offset related to segment registers
	EBP (extended base pointer)	Points to the beginning of the local environment for a function
	ESI (extended source index)	Holds the data source offset in an operation using a memory block
	EDI (extended destination index)	Holds the destination data offset in an operation using a memory block
	ESP (extended stack pointer)	Points to the top of the stack
Special registers		Only used by the CPU
	EFLAGS register; key flags to know are ZF=zero flag; IF=Interrupt enable flag; SF=sign flag	Used by the CPU to track results of logic and the state of processor
	EIP (extended instruction pointer)	Points to the address of the next instruction to be executed

Table 10-4 Categories of Registers

For next class:
get VMs
working.

Coming Soon:
Assembly

Image taken from
IT485 LSN2
slides

SIGSAC – Virtual Machines

- Benefits?
- Virtual Box
- Download linux .iso (we'll use ubuntu for class examples)
- Get it up and running