

Midterm Review

[Data representation](#)

[Binary, Hex, and Decimal Arithmetic](#)

[Memory layout](#)

[Memory Alignment and Structures](#)

[Memory Attacks](#)

[Memory Errors](#)

[Garbage Collection](#)

[x86 Assembly](#)

[Advanced x86 Assembly](#)

[x86 Assembly Procedure Calls](#)

[General Caching](#)

[IO Caching, Coherence](#)

Data Representation

Question: What's the difference between big-endian and little-endian systems?

Exercise: Write how the number 0xDEADBEEF is represented in memory starting at address A in

Big-endian:

A	A+1	A+2	A+3
[0x__	0x__	0x__	0x__]

Little-endian:

A	A+1	A+2	A+3
[0x__	0x__	0x__	0x__]

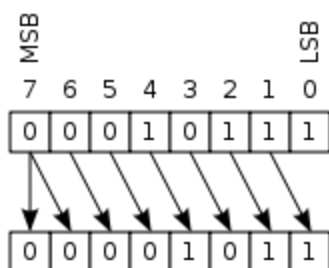
Arithmetic

Question: What do the following operations do to 0011 and 1010:

- Bitwise or |
- Bitwise and &
- Bitwise xor ^

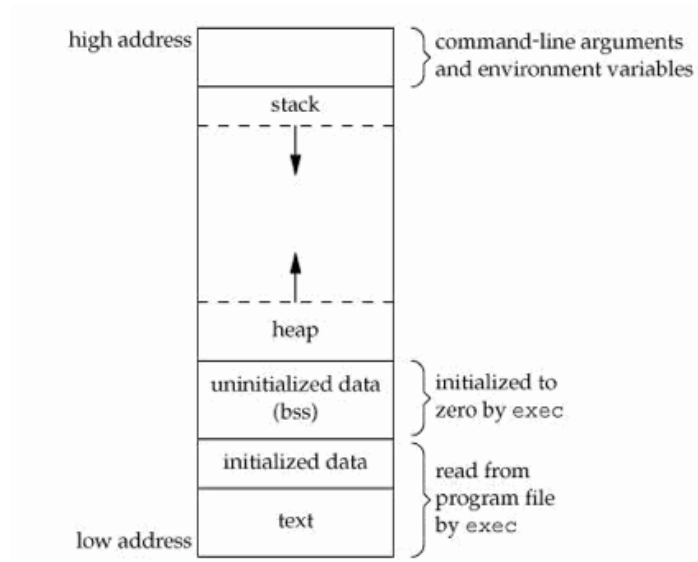
Question: What is ~1100 (not 1100) equal to?

Question: Why are there two types of right shifts but just one type of left shift?



Question: What is the two's complement representation of -8?

Memory Layout



Question: Place the following Qs in the code below into the diagram above.

```
#include <stdlib.h>
#include <stdio.h>

#define GLOBAL_VAR 61      // Q #1
int e = 51;                // Q #2

const char *my_string = "this is fixed";    // Q #3

void help_func(int *c, int *d, int *e) {    // Q #4
    *c = *d * *e; // Q #5
}

int mult(int a, int b) {
    int *answer = malloc(sizeof(int));    // Q #6
    help_func(answer, &a, &b);
    int ans = *answer; // Q #7
    free(answer);
    return ans;
}

int main(void) {
    int result = mult (GLOBAL_VAR, 6);
    printf("The answer is: %d\n", result);
}
```

```
        return 0;
    }
```

Question: Is the following a declaration or a definition?

```
extern int maaaark;
```

```
Float gooldddsteain;
```

```
int* buzzy = 0xBEEEEEEE;
```

Memory Alignment and Structures

Question: Draw pictures and determine the size and alignment of each struct in bytes.

Tip: Remember padding!

```
struct s1 {
    char char1;
    short short1;
    int int1;
    char char2;
}
```

```
struct s2 {
    char char1;
    char char2;
    short short1;
    int int1;
}
```

Memory Attacks

Question: What is stack smashing/buffer overflow?

Question: True or False?

A successful stack smashing attack against a process can cause that process to execute an arbitrary instruction.

Memory Errors

- Memory leaks

- Wild writes
- Double frees/accessing memory after it's been freed
- Dereferencing invalid/inaccessible memory (such as NULL)

Question: For what value of N will the following result in a memory error on a 32-bit system when i=10?

```
int* list = malloc(sizeof(int)*N);
for (int i = 0; i < N; i++) {
    list[i] = i;
}
```

Question: Hello Kitty's Assembly Magic

Unfortunately, Rob broke his compiler and was forced to compile all of his code with the -O0 flag. This means that his code was not optimized in any way by the compiler. Also, he lost all of the C source code because he didn't use GIT. (What a n00b!) Help him recover his C code! Convert the following into C as closely as possible.

1. Hello World!

```
pushq %rbp
movq %rsp, %rbp
// Important Code Below
movl $1337, %eax
movl $22, %ecx
addl %ecx, %eax
popq %rbp
ret
```

2. Turning it up!

Tip: Don't stress if you don't understand the jump commands.

```
pushq %rbp
movq %rsp, %rbp
// Important Code Below
movl $10, %eax
cmpl $4, %eax
jg LABEL1
movl $10, %eax
addl $5, %eax
movl %eax, -8(%rbp)
jmp LABEL2
LABEL1:
```

```

    movl $10, %eax
    subl $5, %eax
    movl %eax, -8(%rbp)
LABEL2:
    movl -8(%rbp), %eax
    movl %eax, -4(%rbp)
    movl -4(%rbp), %eax
    popq %rbp
    ret

```

IO Caching, Coherence

General Statements:

- why do we cache
- spatial locality
- temporal locality
- eviction policies

Question: What is the purpose of a cache? Give an example of a problem introduced by caching.

Question: Briefly define (in your own words) the following terms and give an example?

- Spatial Locality
- Temporal Locality

QUESTION: What is a common situation that would result in a strided access pattern?

Question: Access Patterns

You are given the following access pattern and a 4 slot fully associative cache.

Time	1	2	3	4	5	6	7	8	9	10	11	12	13
Access	a	b	c	d	b	e	a	d	c	f	e	a	d

Cache

Slot	0	1	2	3
Data				

Question: What is in the cache after all 13 accesses are complete if we use the optimal replacement strategy?

Slot	0	1	2	3
Data				

Question: What is the hit ratio?

Question: List the time steps that were cold misses.

Question: If we chose to evict the least recently used (LRU) slot what would be in the cache at the end?

Slot	0	1	2	3
Data				

Question: What is the hit ratio under LRU?

Question: Wrecking Your Hard Drive

You want to copy some music over to your phone. While you wait for the transfer you plan to play “Wrecking Ball” by Miley Cyrus. It’s one of your favorite songs so you know it’s exactly 3 minutes and 41 seconds. If the per request cost is 2.2 microseconds, the per unit cost is 12 nanoseconds, and your music library is 17843750000 bytes, what unit/block size should you use so that the transfer is done just as the song ends?

Tip! Recall:

N = Number of requests in the series

U = Size of each unit in N, for some definition of “unit”

R = Per request cost

K = Per unit cost

C = total cost

$$C = NR + NUK$$

Question: Fill in the blanks with the following words:

main memory
the disk
the processor

1. The processor cache lives in _____. It is a cache for _____.
2. The stdio cache lives in _____. It is a cache for _____.
3. The buffer cache lives in _____. It is a cache for _____.

Question: True, False, or Potato?

1. The stdio library is cache coherent with respect to its primary storage.
2. The buffer cache is cache coherent with respect to its primary storage.
3. The x86 processor cache is cache coherent with respect to its primary storage.
4. There is a single stdio cache that is shared between all applications.
5. There is a single buffer cache that is shared between all applications.
6. Soil Apple

Question: Given that the buffer cache is already caching I/O requests, why does using the stdio cache improve throughput in many cases?

Question: What's up with Rob's Program?

```
#include <stdio.h>
#include <signal.h>

int main(void) {
    printf("now you see me\n");
    printf("now you don't");
    raise(SIGKILL);          // instant-kill
}
```

You compile and run Rob's C program above, and you notice that the program does not output "now you don't" after you run it. Explain what might have caused this behavior so Rob can fix it.

