Computer System Organization [Spring 18, Mu]

R04: String, pointers, and testing

Logistics

- Lab 1 has been graded
  - Max score: 78%
- Many people have received 0 on the first part
  - Code didn’t compile
  - There was the entire grading script provided
    - Please make sure that running tests provided works!
- I will offer regrades on Part 1, fixing the compilation problem for you
  - Regrading will still use your same submission
  - Come see me to write your name and github username
- Don’t commit executables or object files

Exam 1

1. **(C)** A compiler is **machine dependent, and language dependant**
2. **(A)** The number zero is represented specially:

   sign = 0 for positive zero,

   1 for negative zero.

   biased exponent = 0.

   fraction = 0.

Exam 1

3. **(D)**

	32-bit	64-bit
char	1	1
int	4	4
long long	8	8
pointer to long	4	8

Exam 1

4. **(A)** Intel i7 implements x86 instruction set, **cannot** run ARM instructions

5. **(C)**

"The lea instruction places the *address* specified by its first operand into the register specified by its second operand. Note, the *contents* of the memory location are not loaded, only the effective address is computed and placed into the register. This is useful for obtaining a pointer into a memory region or to perform simple arithmetic operations. (http://flint.cs.yale.edu/cs421/papers/x86-asm/asm.html#instructions)"

leaq *address target_register*

leaq 10(%rax, %rax, 2), %rbx

Store %rax + 2*%rax + 10 address into %rbx

Exam 1

6. **(a) 1111 0011 (b) 0xf3 (c) 0x3c**

13 -> 0000 1101
~13 -> 1111 0010
~13 + 1 -> 1111 0011

1111 -> f, 0011 -> 3

Logical right shift do not preserve sign
1111 0011
0011 1100 -> 0x3c

Exam 1

7.

Pass reference to large structs

Data reuse, as in multiple structs could point to the same data without all needing a copy

Refer to data (on heap) that must persist beyond the function it was defined

Exam 1

```
long long foo (long long x) {
    long long sum = 0;
    long long i;
    i = x << 1; // x * 2

    while (i > x) { // i - x > 0
        if (i % 2 == 0) // i & 1 == 0
            sum += 2;
        else
            sum += 3;
        i--;
    }
    return sum;
}
```

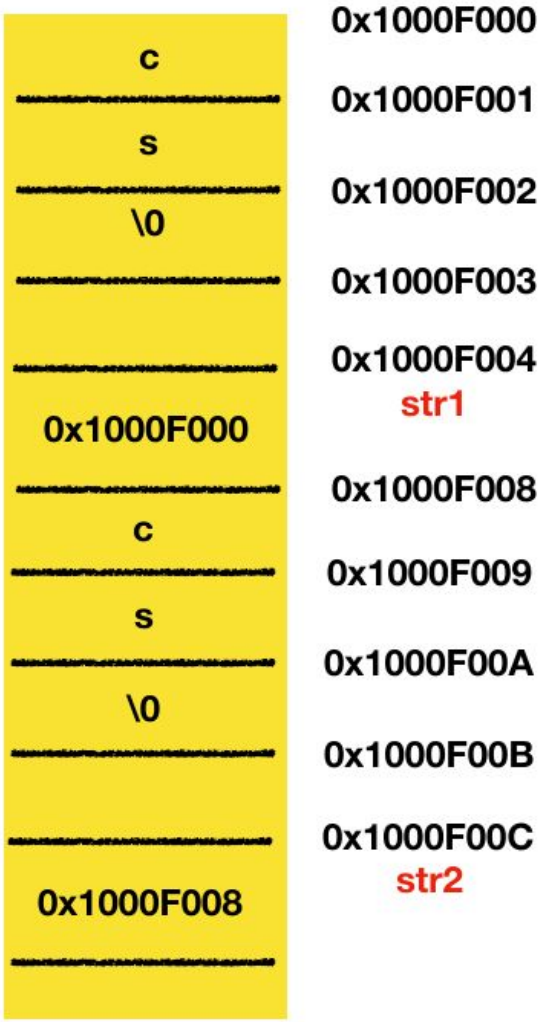
```
foo: xorq %rax, %rax
     movq %rdi, %rbx
     salq $1, %rbx
L3:  cmpq %rdi, %rbx
     jle L0
     testq $1, $rbx
     jne L1
     addq $2, %rax
     jmp L2
L1:  addq $3, %rax
L2:  subq $1, %rbx
     jmp L3
L0:  ret
```

%rax → sum
%rdi → x
%rbx → i
sum = 0;
i = x
i = i << 1
i - x <= 0 ? done

i & 1 ? to else
(if)
  sum += 2
(else)
  sum += 3
i -= 1

Strings in C

```
int main (int argc, char ** argv) {
    char *str1 = "cs";
    char *str2 = "cs";
    printf("%d\n", str1 == str2);
    printf("%d\n", str1[0] == str2[0]);
    printf("%d\n", strcmp(str1, str2));
}
```



Double linked list

```
struct dlist {
    int value;
    struct dlist *prev;
    struct dlist *next;
}
```