CSE 351 15Wi

Midterm Review

Little Things…

- Register are backward compatible
  - Can use the lower part
  - And, of course, change the value for the whole part

Interpreting Assembly

Let `%rax` store x and `%rbx` store y. What does this compute?

```
  mov %rbx, %rcx // ??
  add %rax, %rbx // ??
  je .L1  // ??
  sub %rax, %rcx // ??
  je .L1 // ??
  xor %rax, %rax // ??
  jmp .L2 // ??
L1:
  mov $1, %rax // ??
L2 :
```

Interpreting Assembly

Let `%rax` store x and `%rbx` store y. What does this compute?

```
   mov %rbx, %rcx // %rcx = y
   add %rax, %rbx // ??
   je .L1  // ??
   sub %rax, %rcx // ??
   je .L1 // ??
   xor %rax, %rax // ??
   jmp .L2 // ??
L1:
   mov $1, %rax // ??
L2 :
```

Interpreting Assembly

Let `%rax` store x and `%rbx` store y. What does this compute?

```
  mov %rbx, %rcx // %rcx = y
  add %rax, %rbx // %rbx = y + x
  je .L1  // ??
  sub %rax, %rcx // ??
  je .L1 // ??
  xor %rax, %rax // ??
  jmp .L2 // ??
L1:
  mov $1, %rax // ??
L2 :
```

Interpreting Assembly

Let `%rax` store x and `%rbx` store y. What does this compute?

```
  mov %rbx, %rcx // %rcx = y
  add %rax, %rbx // %rbx = y + x
  je .L1  // jmp to L1 if y + x == 0
  sub %rax, %rcx // ??
  je .L1 // ??
  xor %rax, %rax // ??
  jmp .L2 // ??
L1:
  mov $1, %rax // ??
L2 :
```

Interpreting Assembly

Let `%rax` store x and `%rbx` store y. What does this compute?

```
  mov %rbx, %rcx // %rcx = y
  add %rax, %rbx // %rbx = y + x
  je .L1  // jmp to L1 if y + x == 0
  sub %rax, %rcx // %rcx = y - x
  je .L1 // ??
  xor %rax, %rax // ??
  jmp .L2 // ??
L1:
  mov $1, %rax // ??
L2 :
```

Interpreting Assembly

Let `%rax` store x and `%rbx` store y. What does this compute?

```
  mov %rbx, %rcx // %rcx = y
  add %rax, %rbx // %rbx = y + x
  je .L1  // jmp to L1 if y + x == 0
  sub %rax, %rcx // %rcx = y - x
  je .L1 // jmp to L1 if y - x == 0
  xor %rax, %rax // ??
  jmp .L2 // ??
L1:
  mov $1, %rax // ??
L2 :
```

Interpreting Assembly

Let `%rax` store x and `%rbx` store y. What does this compute?

```
  mov %rbx, %rcx // %rcx = y
  add %rax, %rbx // %rbx = y + x
  je .L1  // jmp to L1 if y + x == 0
  sub %rax, %rcx // %rcx = y - x
  je .L1 // jmp to L1 if y – x == 0
  xor %rax, %rax // %rax = 0
  jmp .L2 // ??
L1:
  mov $1, %rax // ??
L2 :
```

Interpreting Assembly

Let %rax store x and %rbx store y. What does this compute? **|x| == |y|**

```
  mov %rbx, %rcx // %rcx = y
  add %rax, %rbx // %rbx = y + x
  je .L1  // jmp to L1 if y + x == 0
  sub %rax, %rcx // %rcx = y - x
  je .L1 // jmp to L1 if y – x == 0
  xor %rax, %rax // %rax = 0
  jmp .L2 // return 0
L1:
  mov $1, %rax // return 1
L2 :
```

Floating Point Representation

- How is -11.25 is represented in Binary?

- S?
- M in dec?
- M in binary?
- E in dec?
- E in binary?

Pointers

```
int a = 5, b = 15;
int *p1, *p2;

p1 = &a;
p2 = &b;
*p1 = 10;
*p2 = *p1;
p1 = p2;
*p1 = 20;
p1[2] = 10;
```

What are the values of a, b, p1, p2?