CSE 351: Week 5

Tom Bergan, TA

Today

- Midterm Review
    - Past midterm problem: 10au #1
    - Past midterm problem: 10sp #2
    - Past midterm problem: 10au #2
    - Other questions

10au problem #2
What does this function do?

```
mystery: pushl %ebp
         movl %esp, %ebp
         pushl %ebx
         movl 8(%ebp), %ecx
         movl 12(%ebp), %edx
         movl 16(%ebp), %ebx
         movl %ecx, (%edx,%ebx,4)
         movl $0, %eax
         cmpl %ecx, (%edx)
         je L4
         movl $0, %eax
L5:      incl %eax
         cmpl %ecx, (%edx,%eax,4)
         jne L5
L4:      cmpl %ebx, %eax
         setl %al
         movzbl %al, %eax
         popl %ebx
         leave
         ret
```

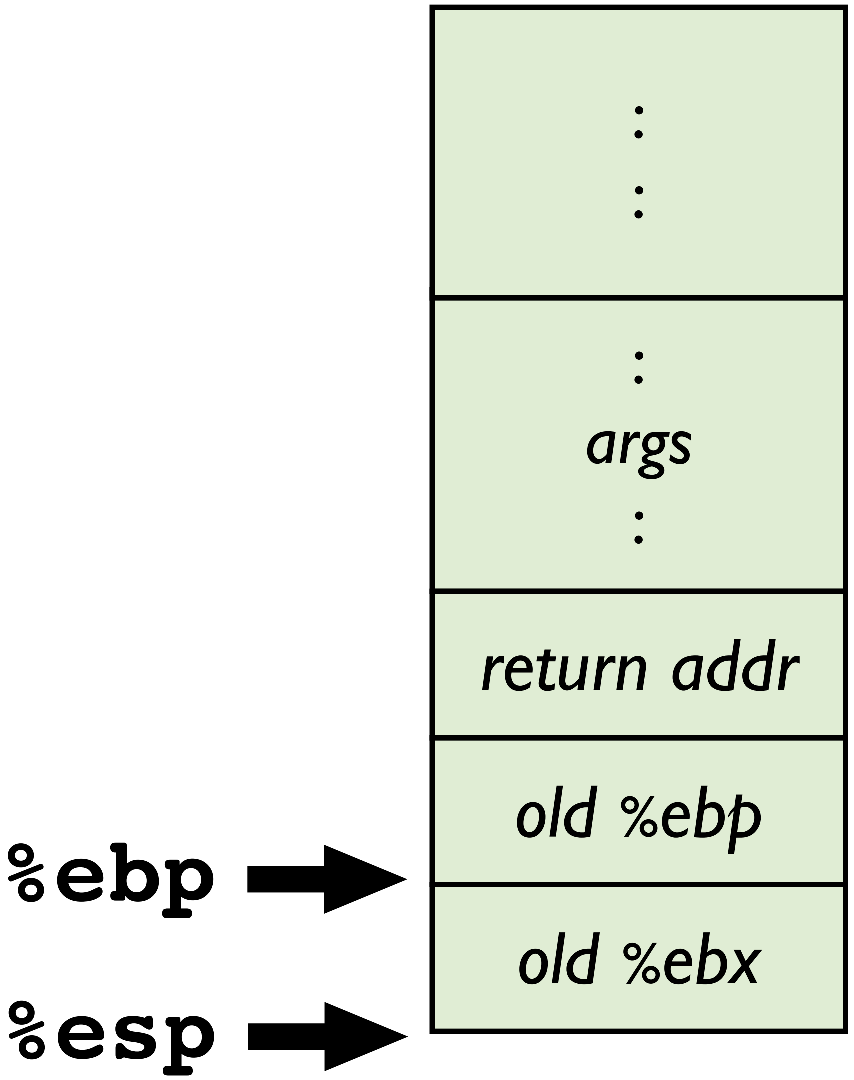
```
mystery: pushl %ebp
         movl %esp, %ebp
         pushl %ebx
         movl 8(%ebp), %ecx
         movl 12(%ebp), %edx
         movl 16(%ebp), %ebx
         movl %ecx, (%edx,%ebx,4)
         movl $0, %eax
         cmpl %ecx, (%edx)
         je L4
         movl $0, %eax

L5:      incl %eax
         cmpl %ecx, (%edx,%eax,4)
         jne L5

L4:      cmpl %ebx, %eax
         setl %al
         movzbl %al, %eax
         popl %ebx
         leave
         ret
```

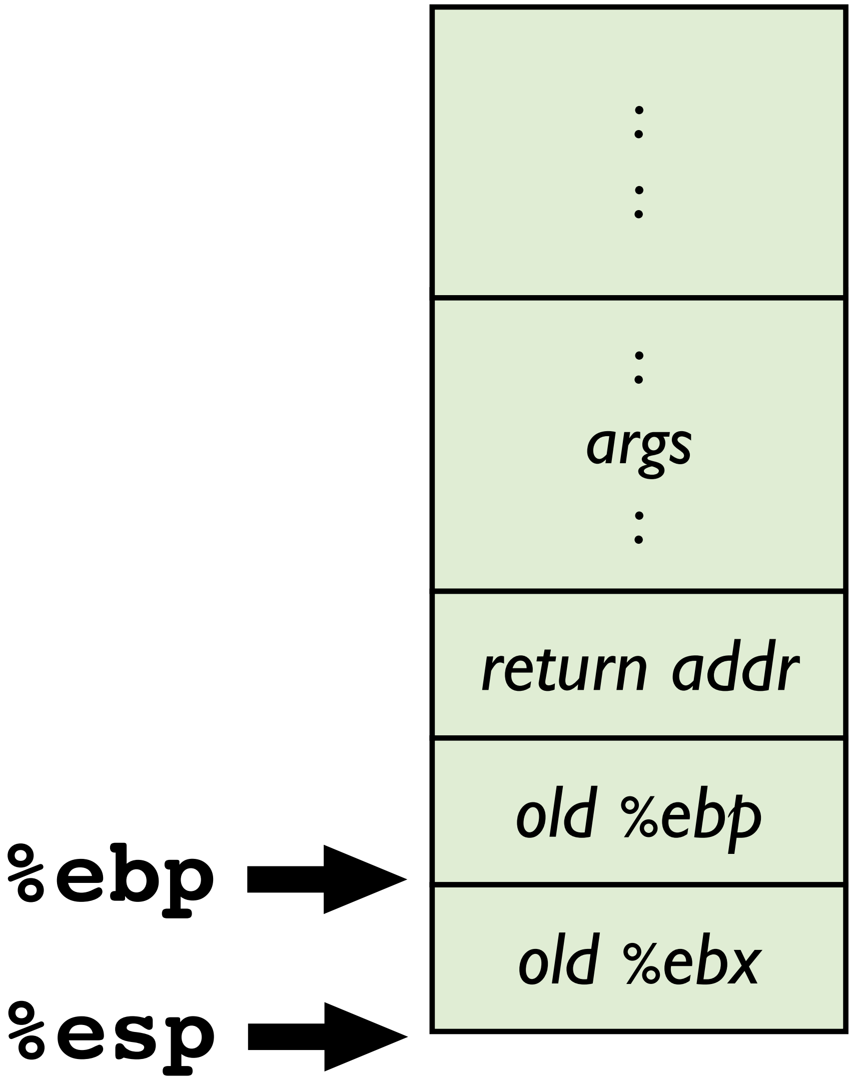
stack setup

The Stack



```
mystery: pushl %ebp
         movl %esp, %ebp
         pushl %ebx
         movl 8(%ebp), %ecx
         movl 12(%ebp), %edx
         movl 16(%ebp), %ebx
         movl %ecx, (%edx,%ebx,4)
         movl $0, %eax
         cmpl %ecx, (%edx)
         je L4
         movl $0, %eax
L5:      incl %eax
         cmpl %ecx, (%edx,%eax,4)
         jne L5
L4:      cmpl %ebx, %eax
         setl %al
         movzbl %al, %eax
         popl %ebx
         leave
         ret
```

The Stack

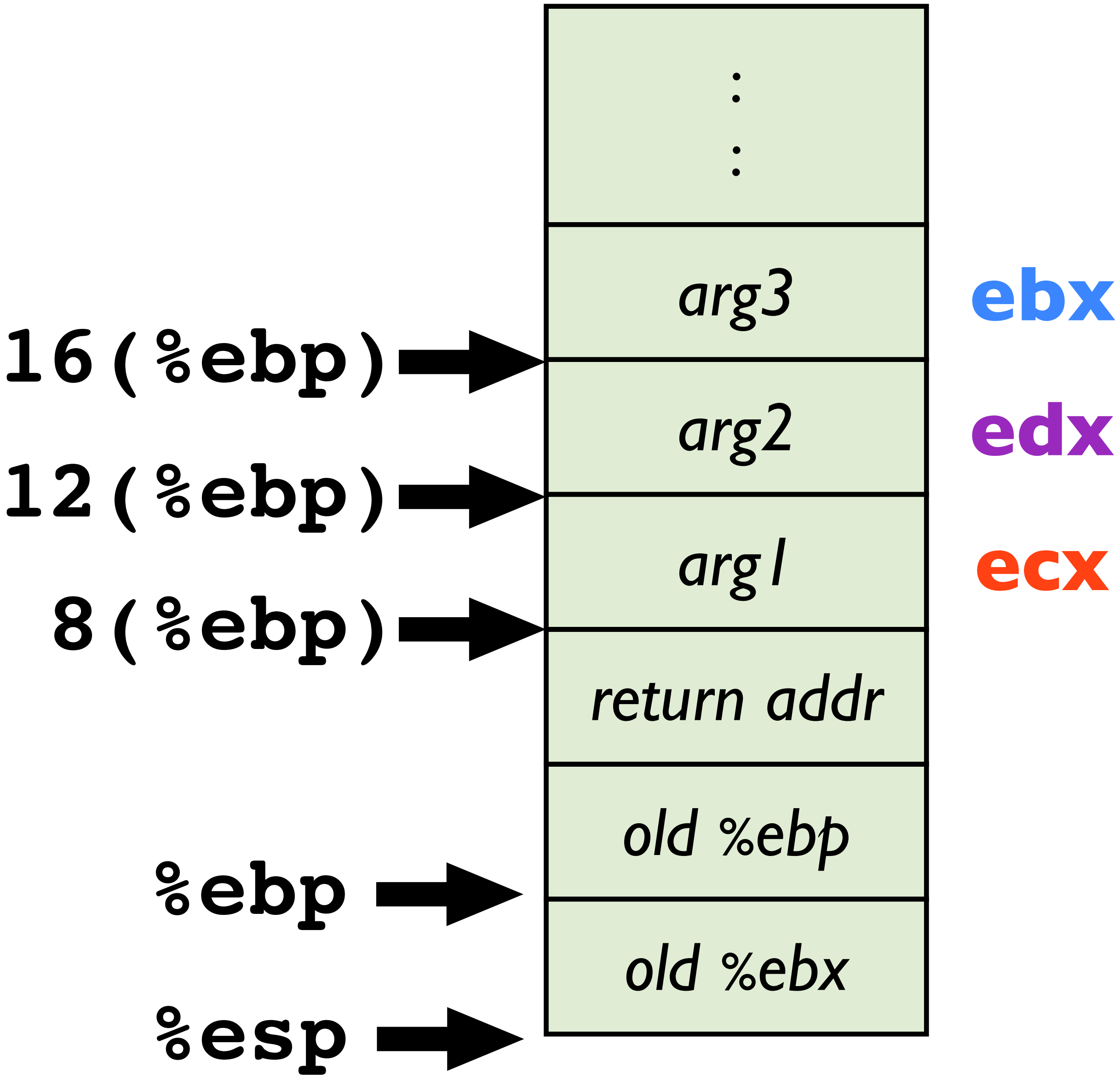


```
mystery: pushl %ebp
         movl %esp, %ebp
         pushl %ebx
         movl 8(%ebp), %ecx
         movl 12(%ebp), %edx
         movl 16(%ebp), %ebx
         movl %ecx, (%edx,%ebx,4)
         movl $0, %eax
         cmpl %ecx, (%edx)
         je L4
         movl $0, %eax

L5:      incl %eax
         cmpl %ecx, (%edx,%eax,4)
         jne L5

L4:      cmpl %ebx, %eax
         setl %al
         movzbl %al, %eax
         popl %ebx
         leave
         ret
```

The Stack



```
mystery: pushl %ebp
         movl %esp, %ebp
         pushl %ebx
         movl 8(%ebp), %ecx
         movl 12(%ebp), %edx
         movl 16(%ebp), %ebx
         movl %ecx, (%edx,%ebx,4)
         movl $0, %eax
         cmpl %ecx, (%edx)
         je L4
         movl $0, %eax
L5:      incl %eax
         cmpl %ecx, (%edx,%eax,4)
         jne L5
L4:      cmpl %ebx, %eax
         setl %al
         movzbl %al, %eax
         popl %ebx
         leave
         ret
```

```
mystery(ecx, edx, ebx) {




    *(edx + 4*ebx) = ecx;
    eax = 0;
```

```
mystery: pushl %ebp
         movl %esp, %ebp
         pushl %ebx
         movl 8(%ebp), %ecx
         movl 12(%ebp), %edx
         movl 16(%ebp), %ebx
         movl %ecx, (%edx,%ebx,4)
         movl $0, %eax
         cmpl %ecx, (%edx)
         je L4
         movl $0, %eax
L5:      incl %eax
         cmpl %ecx, (%edx,%eax,4)
         jne L5
L4:      cmpl %ebx, %eax
         setl %al
         movzbl %al, %eax
         popl %ebx
         leave
         ret
```

```
mystery(ecx, edx, ebx) {





    *(edx + 4*ebx) = ecx;
    eax = 0;
    if (*edx == ecx)
        goto L4;
```

```
mystery: pushl %ebp
         movl %esp, %ebp
         pushl %ebx
         movl 8(%ebp), %ecx
         movl 12(%ebp), %edx
         movl 16(%ebp), %ebx
         movl %ecx, (%edx,%ebx,4)
         movl $0, %eax
         cmpl %ecx, (%edx)
         je L4
         movl $0, %eax

L5:      incl %eax
         cmpl %ecx, (%edx,%eax,4)
         jne L5

L4:      cmpl %ebx, %eax
         setl %al
         movzbl %al, %eax
         popl %ebx
         leave
         ret
```

```
mystery(ecx, edx, ebx) {



    *(edx + 4*ebx) = ecx;
    eax = 0;
    if (*edx == ecx)
        goto L4;
    eax = 0;
```

This is redudant.

(might be trying to fool you?)

We will ignore it.

```
mystery: pushl %ebp
         movl %esp, %ebp
         pushl %ebx
         movl 8(%ebp), %ecx
         movl 12(%ebp), %edx
         movl 16(%ebp), %ebx
         movl %ecx, (%edx,%ebx,4)
         movl $0, %eax
         cmpl %ecx, (%edx)
         je L4
         movl $0, %eax
L5:      incl %eax
         cmpl %ecx, (%edx,%eax,4)
         jne L5
L4:      cmpl %ebx, %eax
         setl %al
         movzbl %al, %eax
         popl %ebx
         leave
         ret
```

```
mystery(ecx, edx, ebx) {

    *(edx + 4*ebx) = ecx;
    eax = 0;
    if (*edx == ecx)
        goto L4;

L5: eax++;
    if (*(edx + 4*eax) != ecx)
        goto L5;
```

```
mystery: pushl %ebp
         movl %esp, %ebp
         pushl %ebx
         movl 8(%ebp), %ecx
         movl 12(%ebp), %edx
         movl 16(%ebp), %ebx
         movl %ecx, (%edx,%ebx,4)
         movl $0, %eax
         cmpl %ecx, (%edx)
         je L4
         movl $0, %eax

L5:      incl %eax
         cmpl %ecx, (%edx,%eax,4)
         jne L5

L4:      cmpl %ebx, %eax
         setl %al
         movzbl %al, %eax
         popl %ebx
         leave
         ret
```

```
mystery(ecx, edx, ebx) {






    *(edx + 4*ebx) = ecx;
    eax = 0;
    if (*edx == ecx)
        goto L4;


L5: eax++;
    if (*(edx + 4*eax) != ecx)
        goto L5;

L4: if (eax < ebx)
        al = 1;
    else
        al = 0;
    eax = al;
```

We can simplify this ...

```
mystery: pushl %ebp
         movl %esp, %ebp
         pushl %ebx
         movl 8(%ebp), %ecx
         movl 12(%ebp), %edx
         movl 16(%ebp), %ebx
         movl %ecx, (%edx,%ebx,4)
         movl $0, %eax
         cmpl %ecx, (%edx)
         je L4
         movl $0, %eax
L5:      incl %eax
         cmpl %ecx, (%edx,%eax,4)
         jne L5
L4:      cmpl %ebx, %eax
         setl %al
         movzbl %al, %eax
         popl %ebx
         leave
         ret
```

```
mystery(ecx, edx, ebx) {





    *(edx + 4*ebx) = ecx;
    eax = 0;
    if (*edx == ecx)
        goto L4;

L5: eax++;
    if (*(edx + 4*eax) != ecx)
        goto L5;
L4: if (eax < ebx)
        eax = 1;
    else
        eax = 0;
```

```
mystery: pushl %ebp
         movl %esp, %ebp
         pushl %ebx
         movl 8(%ebp), %ecx
         movl 12(%ebp), %edx
         movl 16(%ebp), %ebx
         movl %ecx, (%edx,%ebx,4)
         movl $0, %eax
         cmpl %ecx, (%edx)
         je L4
         movl $0, %eax
L5:      incl %eax
         cmpl %ecx, (%edx,%eax,4)
         jne L5
L4:      cmpl %ebx, %eax
         setl %al
         movzbl %al, %eax
         popl %ebx
         leave
         ret
```

```
mystery(ecx, edx, ebx) {






    *(edx + 4*ebx) = ecx;
    eax = 0;
    if (*edx == ecx)
        goto L4;


L5: eax++;
    if (*(edx + 4*eax) != ecx)
        goto L5;
L4: if (eax < ebx)
        eax = 1;
    else
        eax = 0;
    return eax;
}
```

What they gave us

```
int mystery(int x, int A[], int n) {
  int k, result;

  ________________________________________ ;

  k = 0;

  while ( ________________________________ ) {

    ____________________;

  }

  if ( ___________________________ ) {

    result = 1;

  } else {

    result = 0;

  }

  return result;

}
```

we know the names and types of the args!

Our C code

```
mystery(ecx, edx, ebx) {
        *(edx + 4*ebx) = ecx;
        eax = 0;

        if (*edx == ecx)
                goto L4;
L5:     eax++;
        if (*(edx + 4*eax) != ecx)
                goto L5;

L4:     if (eax < ebx)
                eax = 1;
        else
                eax = 0;
        return eax;
}
```

How do we translate this?

... array access!

edx = base of array
ebx = index of array
4 = sizeof an element

```
             ecx    edx      ebx
int mystery(int x, int A[], int n) {
    *(edx + 4*ebx) = x;
    eax = 0;

    if (*edx == x)
        goto L4;
L5: eax++;
    if (*(edx + 4*eax) != x)
        goto L5;

L4: if (eax < n)
        eax = 1;
    else
        eax = 0;
    return eax;
}
```

Need to translate pointer arithmetic!

How do we translate this?

A[n]

What about this?

A[0]

What about this?

A[eax]

```
                             ecx       edx        ebx
int mystery(int x, int A[], int n) {
    *(edx + 4*ebx) = x;
    eax = 0;

    if (*edx == x)
        goto L4;
L5: eax++;
    if (*(edx + 4*eax) != x)
        goto L5;

L4: if (eax < n)
        eax = 1;
    else
        eax = 0;
    return eax;
}
```

What does this look like? →

do .. while() loop

```
int mystery(int x, int A[], int n) {
    A[n] = x;
    eax = 0;

    if (A[0] == x)
        goto L4;
L5: eax++;
    if (A[eax] != x)
        goto L5;

L4: if (eax < n)
        eax = 1;
    else
        eax = 0;
    return eax;
}
```

What does this look like?

while() loop

```
int mystery(int x, int A[], int n) {
    A[n] = x;
    eax = 0;

    if (A[0] == x)
        goto L4;
    do {
        eax++;
    } while (A[eax] != x);

L4: if (eax < n)
        eax = 1;
    else
        eax = 0;
    return eax;
}
```

```
int mystery(int x, int A[], int n) {
    A[n] = x;
    eax = 0;

    while (A[eax] != x) {
        eax++;
    }

    if (eax < n)
        eax = 1;
    else
        eax = 0;
    return eax;
}
```

What they gave us

```
int mystery(int x, int A[], int n) {
  int k, result;

  _______________________________________ ;
  k = 0;
  while ( ________________________________ ) {

    ____________________;
  }
  if ( ___________________________ ) {
    result = 1;
  } else {
    result = 0;
  }
  return result;
}
```

Our C code

```
int mystery(int x, int A[], int n)
{

    A[n] = x;
    eax = 0;
    while (A[eax] != x) {

        eax++;
    }
    if (eax < n)
        eax = 1;
    else
        eax = 0;
    return eax;
}
```

What they gave us

```
int mystery(int x, int A[], int n) {
  int k, result;

  _______________________________________ ;
  k = 0;
  while ( ________________________________ ) {

    ____________________;
  }
  if ( ___________________________ ) {
    result = 1;
  } else {
    result = 0;
  }
  return result;
}
```

Our C code

```
int mystery(int x, int A[], int n)
{

    A[n] = x;
    k = 0;
    while (A[k] != x) {

        k++;
    }
    if (k < x)
        result = 1;
    else
        result = 0;
    return result;
}
```