# Announcements

- Assignment 2 due 5 p.m. today
- Assignment 3 to be posted today
- No class on Monday, October 13 (Trinity Day)
- Quiz on Wednesday, October 15

Lecture 15

# Arithmetic Operations in IA-32

CPSC 275
Introduction to Computer Systems

# Address Computation

- **`leal` *Src*, *Dst***
  - "**l**oad **e**ffective **a**ddress"
  - *Src* is address mode expression
  - Set *Dst* to address denoted by expression

- Uses
  - Computing addresses without a memory reference
    - e.g., translation of **`p = &x[i];`**
  - Computing arithmetic expressions of the form

    **`x + k*y`**

    e.g., Suppose **`%edx`** has the value y

    **`leal 7(%edx,%edx,4)`** computes **`7 + 5*y`**

# Arithmetic Operations

- Two operand instructions:

| Format | | Computation |
|---|---|---|
| `addl` | *Src,Dest* | Dest = Dest + Src |
| `subl` | *Src,Dest* | Dest = Dest – Src |
| `imull` | *Src,Dest* | Dest = Dest * Src |

- Watch out for argument order!
- No distinction between `signed` and `unsigned int` (why?)

- One operand instructions:

| Format | | Computation |
|---|---|---|
| `incl` | *Dest* | *Dest = Dest + 1* |
| `decl` | *Dest* | *Dest = Dest – 1* |
| `negl` | *Dest* | *Dest = – Dest* |

# Logical Operations

| *Format* | | *Computation* |
|---|---|---|
| `xorl` | *Src,Dest* | Dest = Dest ^ Src |
| `andl` | *Src,Dest* | Dest = Dest & Src |
| `orl` | *Src,Dest* | Dest = Dest \| Src |
| `notl` | *Dest* | Dest = ~Dest |

# Arithmetic Expression Example

```
int arith(int x, int y, int z) {
  int t1 = x + y;
  int t2 = z + t1;
  int t3 = x + 4;
  int t4 = y * 48;
  int t5 = t3 + t4;
  int rval = t2 * t5;
  return rval;
}
```

# Arithmetic Expression Example

```
int arith(int x, int y, int z) {
  int t1 = x + y;
  int t2 = z + t1;
  int t3 = x + 4;
  int t4 = y * 48;
  int t5 = t3 + t4;
  int rval = t2 * t5;
  return rval;
}
```

| Register | Use(s) |
|---|---|
| `%edi` | Argument **`x`** |
| `%esi` | Argument **`y`** |
| `%edx` | Argument **`z`** |

# Arithmetic Expression Example

```
int arith(int x, int y, int z) {
  int t1 = x + y;
  int t2 = z + t1;
  int t3 = x + 4;
  int t4 = y * 48;
  int t5 = t3 + t4;
  int rval = t2 * t5;
  return rval;
}
```

| Register | Use(s) |
|---|---|
| %edi | Argument x |
| %esi | Argument y |
| %edx | Argument z |

```
arith:
  leal    (%edi,%esi), %eax    # t1
```

# Arithmetic Expression Example

```
int arith(int x, int y, int z) {
  int t1 = x + y;
  int t2 = z + t1;
  int t3 = x + 4;
  int t4 = y * 48;
  int t5 = t3 + t4;
  int rval = t2 * t5;
  return rval;
}
```

| Register | Use(s) |
|---|---|
| %edi | Argument x |
| %esi | Argument y |
| %edx | Argument z |

```
arith:
  leal    (%edi,%esi), %eax    # t1
  addl    %edx, %eax           # t2
```

# Arithmetic Expression Example

```
int arith(int x, int y, int z) {
  int t1 = x + y;
  int t2 = z + t1;
  int t3 = x + 4;
  int t4 = y * 48;
  int t5 = t3 + t4;
  int rval = t2 * t5;
  return rval;
}
```

| Register | Use(s) |
|---|---|
| %edi | Argument x |
| %esi | Argument y |
| %edx | Argument z |

```
arith:
  leal    (%edi,%esi), %eax    # t1
  addl    %edx, %eax           # t2
  leal    (%esi,%esi,2), %edx  # 3*y
  sall    $4, %edx             # t4
```

# Arithmetic Expression Example

```
int arith(int x, int y, int z) {
  int t1 = x + y;
  int t2 = z + t1;
  int t3 = x + 4;
  int t4 = y * 48;
  int t5 = t3 + t4;
  int rval = t2 * t5;
  return rval;
}
```

| Register | Use(s) |
|---|---|
| %edi | Argument **x** |
| %esi | Argument **y** |
| %edx | Argument **z** |

```
arith:
  leal    (%edi,%esi), %eax    # t1
  addl    %edx, %eax           # t2
  leal    (%esi,%esi,2), %edx  # 3*y
  sall    $4, %edx             # t4
  leal    4(%edi,%edx), %ecx   # t5
```

# Arithmetic Expression Example

```
int arith(int x, int y, int z) {
  int t1 = x + y;
  int t2 = z + t1;
  int t3 = x + 4;
  int t4 = y * 48;
  int t5 = t3 + t4;
  int rval = t2 * t5;
  return rval;
}
```

| Register | Use(s) |
|---|---|
| %edi | Argument x |
| %esi | Argument y |
| %edx | Argument z |

```
arith:
  leal    (%edi,%esi), %eax    # t1
  addl    %edx, %eax           # t2
  leal    (%esi,%esi,2), %edx  # 3*y
  sall    $4, %edx             # t4
  leal    4(%edi,%edx), %ecx   # t5
  imull   %ecx, %eax           # rval
```

# Arithmetic Expression Example

```
int arith(int x, int y, int z) {
  int t1 = x + y;
  int t2 = z + t1;
  int t3 = x + 4;
  int t4 = y * 48;
  int t5 = t3 + t4;
  int rval = t2 * t5;
  return rval;
}
```

| Register | Use(s) |
|---|---|
| **%edi** | Argument **x** |
| **%esi** | Argument **y** |
| **%edx** | Argument **z** |

```
arith:
  leal    (%edi,%esi), %eax    # t1
  addl    %edx, %eax           # t2
  leal    (%esi,%esi,2), %edx  # 3*y
  sall    $4, %edx             # t4
  leal    4(%edi,%edx), %ecx   # t5
  imull   %ecx, %eax           # rval
  ret
```

return value

# Arithmetic Expression Example

```
int arith(int x, int y, int z) {
  int t1 = x + y;
  int t2 = z + t1;
  int t3 = x + 4;
  int t4 = y * 48;
  int t5 = t3 + t4;
  int rval = t2 * t5;
  return rval;
}
```

| Register | Use(s) |
|---|---|
| %edi | Argument **x** |
| %esi | Argument **y** |
| %edx | Argument **z** |
| %eax | **t1**, **t2**, **rval** |
| %edx | **t4** |
| %ecx | **t5** |

```
arith:
  leal    (%edi,%esi), %eax    # t1
  addl    %edx, %eax           # t2
  leal    (%esi,%esi,2), %edx  # 3*y
  sall    $4, %edx             # t4
  leal    4(%edi,%edx), %ecx   # t5
  imull   %ecx, %eax           # rval
  ret
```