More x86 Code

Computer Systems Sections 3.3-3.6

See Also:

http://www.cs.binghamton.edu/~tbartens/CS220_Spring_2016/examples/xmp_x86

If/Then/Else

```
int myfunc(int a,int b) {
        int c;
        if (a>b) c = a;
        else c = b;
        return c;
}
```

```
        pushl   %ebp
        movl    %esp, %ebp
        subl    $16, %esp
        movl    8(%ebp), %eax
        cmpl    12(%ebp), %eax
        jle     .L2
        movl    8(%ebp), %eax
        movl    %eax, -4(%ebp)
        jmp     .L3
.L2:
        movl    12(%ebp), %eax
        movl    %eax, -4(%ebp)
.L3:
        movl    -4(%ebp), %eax
        leave
        ret
```

Memory	
b	x0000 0004
a	x0000 0003
(%ebp)	
c	xdead beef

While Loops

```
int myfunc(int a,int b) {
        int c;
        while(a<10) {
                c+=b;
                a--;
        }
        return c;
}
```

gcc -m32 -O0 -S prog.c

```
int myfunc(int a,int b) {
        int c;
        while(a<10) {
                c+=b;
                a--;
        }
        return c;
}
```

```
myfunc:
        pushl   %ebp
        movl    %esp, %ebp
        subl    $16, %esp
        jmp     .L2
.L3:
        movl    12(%ebp), %eax
        addl    %eax, -4(%ebp)
        subl    $1, 8(%ebp)
.L2:
        cmpl    $9, 8(%ebp)
        jle     .L3
        movl    -4(%ebp), %eax
        leave
        ret
```

While Condition

```
int myfunc(int a,int b) {
        int c;
        while(a<10) {
                c+=b;
                a--;
        }
        return c;
}
```

```
myfunc:
        pushl   %ebp
        movl    %esp, %ebp
        subl    $16, %esp
        jmp     .L2
.L3:
        movl    12(%ebp), %eax
        addl    %eax, -4(%ebp)
        subl    $1, 8(%ebp)
.L2:
        cmpl    $9, 8(%ebp)
        jle     .L3
        movl    -4(%ebp), %eax
        leave
        ret
```

Memory	
b	x0000 0004
a	x0000 0003
(%ebp)	
c	xdead beef

Loop Body

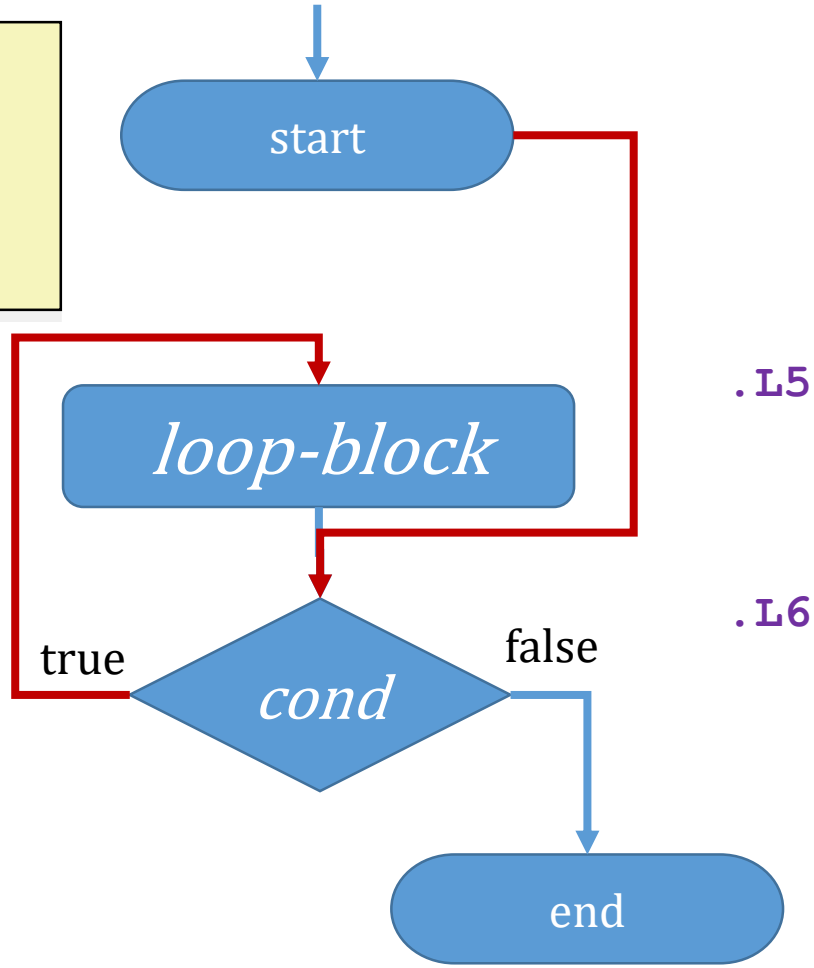
```
int myfunc(int a,int b) {
        int c;
        while(a<10) {
                c+=b;
                a--;
        }
        return c;
}
```

```
myfunc:
        pushl   %ebp
        movl    %esp, %ebp
        subl    $16, %esp
        jmp     .L2
.L3:
        movl    12(%ebp), %eax
        addl    %eax, -4(%ebp)
        subl    $1, 8(%ebp)
.L2:
        cmpl    $9, 8(%ebp)
        jle     .L3
        movl    -4(%ebp), %eax
        leave
        ret
```

Memory	
b	x0000 0004
a	x0000 0003
(%ebp)	
c	xdead beef

Translating C to x86: while

```
while ( cond ) {
        loop-block
}
```




```
        …
        jmp L6

.L5:
        …           ; loop block

.L6:
        cmpl    …; condition
        jxx     .L5

        …
```

For Loops

```
int myfunc(int a,int b) {
        int c;
        for(c=0;c<a;c++) {
                b=b+a;
        }
        return b;
}
```

gcc -m32 -O0 -S prog.c

```
int myfunc(int a,int b) {
        int c;
        for(c=0;c<a;c++) {
                b=b+a;
        }
        return b;
}
```

```
myfunc:
        pushl   %ebp
        movl    %esp, %ebp
        subl    $16, %esp
        movl    $0, -4(%ebp)
        jmp     .L2
.L3:
        movl    8(%ebp), %eax
        addl    %eax, 12(%ebp)
        addl    $1, -4(%ebp)
.L2:
        movl    -4(%ebp), %eax
        cmpl    8(%ebp), %eax
        jl      .L3
        movl    12(%ebp), %eax
        leave
        ret
```

For Statement

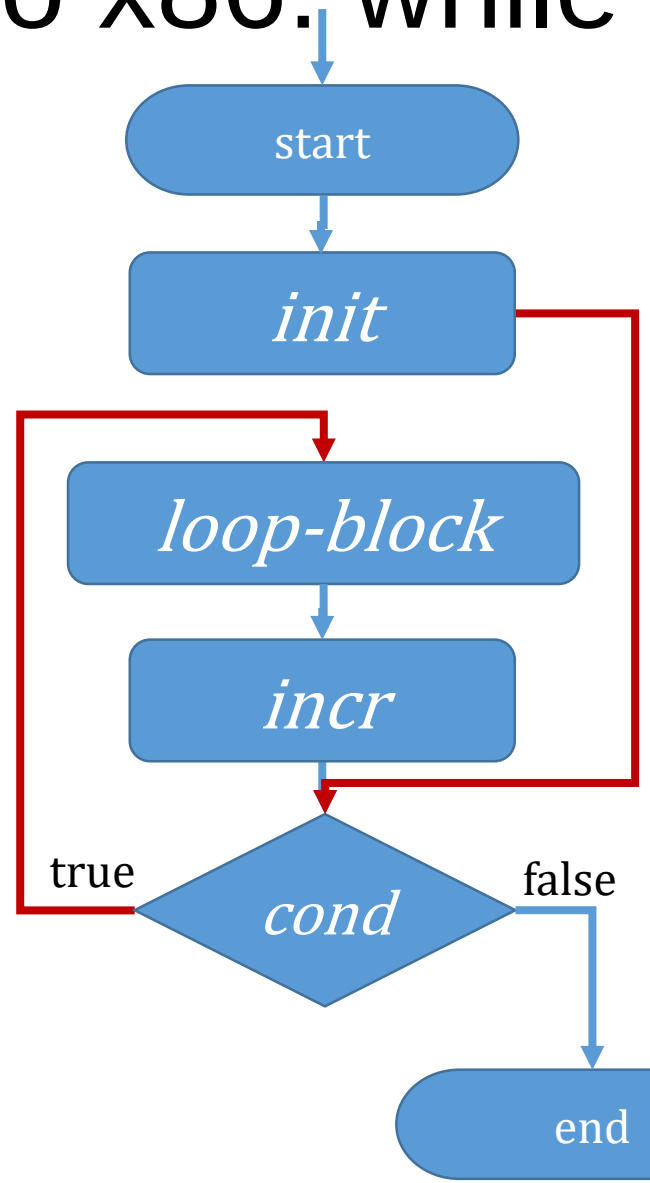
```
int myfunc(int a,int b) {
        int c;
        for(c=0;c<a;c++) {
                b=b+a;
        }
        return b;
}
```

```
myfunc:
        pushl   %ebp
        movl    %esp, %ebp
        subl    $16, %esp
        movl    $0, -4(%ebp)
        jmp     .L2
.L3:
        movl    8(%ebp), %eax
        addl    %eax, 12(%ebp)
        addl    $1, -4(%ebp)
.L2:
        movl    -4(%ebp), %eax
        cmpl    8(%ebp), %eax
        jl      .L3
        movl    12(%ebp), %eax
        leave
        ret
```

Memory	
b	x0000 0004
a	x0000 0003
(%ebp)	
c	xdead beef

Translating C to x86: while

```
for(init;cond;incr) {
        loop-block
}
```




```
        ...
        ...     ; init
        jmp L6
.L5:
        ...     ; loop block

        ...     ; incr

.L6:
        cmpl    ...; condition
        jxx     .L5

        ...
```

Switch Statements

```
int myfunc(int a,int b) {
        int c;
        switch(a) {
                case 10 : c=a; break;
                case 11: b++;
                case 12:
                case 13: c=a+b; break;
                case 14: c=a*b; break;
                case 15: c=a+2; break;
                default: c=0;
        }
        return c;
}
```

gcc -m32 -O0 -S progSwitch.c

```
int myfunc(int a,int b) {
    int c;
    switch(a) {
        case 10 : c=a; break;
        case 11: b++;
        case 12:
        case 13: c=a+b; break;
        case 14: c=a*b; break;
        case 15: c=a+2; break;
        default: c=0;
    }
    return c;
}
```

```
        movl    8(%ebp), %eax
        subl    $10, %eax
        cmpl    $5, %eax
        ja      .L2
        movl    .L8(,%eax,4),
                %eax
        jmp     *%eax
.L8:
        .long   .L3
        .long   .L4
        .long   .L5
        .long   .L5
        .long   .L6
        .long   .L7
.L3:
        movl    8(%ebp), %eax
        movl    %eax, -4(%ebp)
        jmp     .L9
.
```

```
.L4:
        addl    $1, 12(%ebp)
.L5:
        movl    12(%ebp), %eax
        movl    8(%ebp), %edx
        leal    (%edx,%eax), %eax
        movl    %eax, -4(%ebp)
        jmp     .L9
.L6:
        movl    8(%ebp), %eax
        imull   12(%ebp), %eax
        movl    %eax, -4(%ebp)
        jmp     .L9
.L7:
        movl    8(%ebp), %eax
        addl    $2, %eax
        movl    %eax, -4(%ebp)
        jmp     .L9
.L2:
        movl    $0, -4(%ebp)
.L9:
```

Case Blocks

```
int myfunc(int a,int b) {
    int c;
    switch(a) {
        case 10 : c=a; break;
        case 11: b++;
        case 12:
        case 13: c=a+b; break;
        case 14: c=a*b; break;
        case 15: c=a+2; break;
        default: c=0;
    }
    return c;
}
```

```
movl    8(%ebp), %eax
subl    $10, %eax
cmpl    $5, %eax
ja      .L2
movl    .L8(,%eax,4), %eax
jmp     *%eax
.L8:
.long   .L3
.long   .L4
.long   .L5
.long   .L5
.long   .L6
.long   .L7
```

```
.L3:
    movl    8(%ebp), %eax
    movl    %eax, -4(%ebp)
    jmp     .L9
```

```
.L4:
    addl    $1, 12(%ebp)
.L5:
    movl    12(%ebp), %eax
    movl    8(%ebp), %edx
    leal    (%edx,%eax), %eax
    movl    %eax, -4(%ebp)
    jmp     .L9
.L6:
    movl    8(%ebp), %eax
    imull   12(%ebp), %eax
    movl    %eax, -4(%ebp)
    jmp     .L9
.L7:
    movl    8(%ebp), %eax
    addl    $2, %eax
    movl    %eax, -4(%ebp)
    jmp     .L9
.L2:
    movl    $0, -4(%ebp)
.L9:
```

Switch "Jump Table"

```
int myfunc(int a,int b) {
    int c;
    switch(a) {
        case 10 : c=a; break;
        case 11: b++;
        case 12:
        case 13: c=a+b; break;
        case 14: c=a*b; break;
        case 15: c=a+2; break;
        default: c=0;
    }
    return c;
}
```

```
        movl    8(%ebp), %eax
        subl    $10, %eax
        cmpl    $5, %eax
        ja      .L2
        movl    .L8(,%eax,4),
%eax
        jmp     *%eax
        .L8:
        .long   .L3
        .long   .L4
        .long   .L5
        .long   .L5
        .long   .L6
        .long   .L7
.L3:
        movl    8(%ebp), %eax
        movl    %eax, -4(%ebp)
        jmp     .L9
.
```

```
.L4:
        addl    $1, 12(%ebp)
.L5:
        movl    12(%ebp), %eax
        movl    8(%ebp), %edx
        leal    (%edx,%eax), %eax
        movl    %eax, -4(%ebp)
        jmp     .L9
.L6:
        movl    8(%ebp), %eax
        imull   12(%ebp), %eax
        movl    %eax, -4(%ebp)
        jmp     .L9
.L7:
        movl    8(%ebp), %eax
        addl    $2, %eax
        movl    %eax, -4(%ebp)
        jmp     .L9
.L2:
        movl    $0, -4(%ebp)
.L9:
```

L8 @ x080484d0 – "Jump Table"

Location	Label	Index	"a" value	Value	Points to
x080484d0	.L3	0	10	x080483ae	case 10
x080484d4	.L4	1	11	x080483b6	case 11
x080484d8	.L5	2	12	x080483ba	case 13
x080484dc	.L5	3	13	x080483ba	case 13
x080484e0	.L6	4	14	x080483c8	case 14
x080484e4	.L7	5	15	x080483d4	case 15

Evaluating the Switch

```
int myfunc(int a,int b) {
    int c;
    switch(a) {
        case 10 : c=a; break;
        case 11: b++;
        case 12:
        case 13: c=a+b; break;
        case 14: c=a*b; break;
        case 15: c=a+2; break;
        default: c=0;
    }
    return c;
}
```

```
        movl    8(%ebp), %eax
        subl    $10, %eax
        cmpl    $5, %eax
        ja      .L2
        movl    .L8(,%eax,4), %eax
        jmp     *%eax
.L8:
        .long   .L3
        .long   .L4
        .long   .L5
        .long   .L5
        .long   .L6
        .long   .L7
.L3:
        movl    8(%ebp), %eax
        movl    %eax, -4(%ebp)
        jmp     .L9
.L4:
        addl    $1, 12(%ebp)
.L5:
        movl    12(%ebp), %eax
        movl    8(%ebp), %edx
        leal    (%edx,%eax), %eax
        movl    %eax, -4(%ebp)
        jmp     .L9
.L6:
        movl    8(%ebp), %eax
        imull   12(%ebp), %eax
        movl    %eax, -4(%ebp)
        jmp     .L9
.L7:
        movl    8(%ebp), %eax
        addl    $2, %eax
        movl    %eax, -4(%ebp)
        jmp     .L9
.L2:
        movl    $0, -4(%ebp)
.L9:
```

Indirect Jump

jmp *<reference>

- <reference> : A location/register that contains the target
- May also see table addressing notation: jmp *.L4(,%eax,4)
  - Use %eax as index into a table starting at L4
  - Each entry in the table is 4 bytes wide
  - Use the %eaxth entry as the target for the jump