Computer Systems
Lecture 12

Overview

- Controlling program flow: Loops
- Loops with additional tests
- Implementing higher-order constructs: conditional statements
- Implementing higher-order constructs: the *for* statement
- Implementing higher-order constructs: the *while* statement
- Higher-order constructs:
  do-while statement
- *switch-case* statement

Controlling program flow: Loops

- Simple loop:

`loop <label>`

Automatically decrements `ECX`

  when `ECX = 0`, the loop ends

  when `ECX` is not `0` it jumps to `<label>`

A simple loop example

- Write a LOOP instruction: Repeat by counting down from `200` to `0` and do some task in the loop.

```
          mov ecx, 200  ; set counter
next:…                  ; do the task here
          …
          loop next     ; jump to the label 'next'
          …             ; continue after looping
```

Another example – loop upon two conditions

- LOOPNE instruction
- While EAX is not equal to EBX AND not 200 times yet

```
    mov ecx,200     ; Set counter
next: …             ; Set label
    …               ; Do something
    cmp eax,ebx     ; Are EAX and EBX the
                    ; same? Or 200 times already?
    loopne next     ; No? Go to next.
    …               ; Yes? Continue.
```

Implementing higher-order constructs: conditional statements

- In Java:

```
if (c > 0)
  pos = pos + c;
else
  neg = neg + c;
```

- Equivalent in the assembly code:

```
          mov eax,c
          cmp eax,0
          jg positive
negative:add neg,eax
          jmp endif
positive:add pos,eax
endif:…
```

Implementing higher-order constructs: conditional statements

- In Java:

```
if (c > 0)
  pos = pos + c;
else
  neg = neg + c;
```

- Equivalent in the assembly code:

```
          mov eax,c
          cmp eax,0
          jle negative
positive:add pos,eax
          jmp endif
negative:add neg,eax
endif:…
```

Implementing higher-order constructs: the *for* statement

- In Java:

```
for (int x = 0;
  x < 10; x++)
{
  y = y +x;
}
```

- Equivalent in the assembly code:

```
          mov eax,0
for_loop:add y,eax
          inc eax
          cmp eax,10
          jl for_loop
```

Problem of this implementation?

Implementing *for* statement : Exercise

- In Java:

```
for (int x = 3;
  x < 20; x=x+2)
{
  y = y +x;
}
```

Implementing higher-order constructs: the *while* statement

- In Java:

```
while (fib2 < 1000)
{
  fib0 = fib1;
  fib1 = fib2;
  fib2 = fib1 + fib0;
}
```

- Equivalent in the assembly code:

```
while:mov eax,fib2
    cmp eax,1000
    jge end_while
    mov eax,fib1
    mov fib0,eax
    mov eax,fib2
    mov fib1,eax
    add eax, fib0
    mov fib2,eax
    jmp while
end_while: …
```

do-while statement : Exercise

- In Java:

```
do{
  fib0 = fib1;
  fib1 = fib2;
  fib2 = fib1 + fib0;
}while (fib2 < 1000)
```

- Equivalent in the assembly code:

Switch-Case statement

- In Java:

```
switch (num){
  case 1:  … ;
     break;
  case 2:  … ;
     break;
}
```

- Equivalent in the assembly code:

```
mov eax,num
    cmp eax,1
    je case_1
    cmp eax,2
    je case_2
    jmp end_case
case_1:…
    jmp end_case
case_2:…
end_case:…
```

Implementing ‘loop’

- Loop in an assembly code:

```
    mov ecx,200
next:…
    …
    …
    loop next
```

- Can we do without ‘loop’?
- Hint: use ‘cmp’, ‘dec’, ‘jne’
- Equivalent without loop construction:

```
  mov ecx,200
next:…
  …
  …
  dec ecx
  cmp ecx,0
  jne next
```

- Q. Conditional jumps in assembly can be used to implement HLL constructs like while, for and switch. (T or F)

- Q. ‘loop’ instruction in assembly has a branching effect based upon the value of decremented ECX register. (T or F)

- Q. Explain what ‘LOOPNE label’ does.

- Q. Explain what the following instructions do.

```
CMP EAX, EBX
LOOPNE label
```