

CSC231 — Assembly

Week #8 — Fall 2017

Dominique Thiébaud
dthiebaut@smith.edu

Useful Property

00000000
00000001
00000010
00000011
00000100
...
01111110
01111111
10000000
10000001
10000010
...
11111110
11111111

b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0

-2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0



weights of bits for 2's complement

Exercises on Signed Numbers



http://www.science.smith.edu/dftwiki/index.php/CSC231_Exercises_on_Signed_Numbers

Useful On-Line Tool

<http://www.exploringbinary.com/twos-complement-converter/>

The screenshot shows the website's navigation menu with links for Home, Converters/Calculators, Topics, Sitemap, About, and Contact. The main heading is "Decimal/Two's Complement Converter". Below it, there are two sections: "Decimal to Two's Complement" and "Two's Complement to Decimal".

Decimal to Two's Complement
Enter a decimal integer (e.g., -2013) (no commas or spaces)
-1
Converts to this two's complement binary integer:
1111111111111111
Buttons: Convert, Clear
Options:
• Number of bits: 16

Two's Complement to Decimal
Enter a two's complement binary integer (e.g., 00010110) (no commas or spaces)
1000000000000001

On the right side, there is a search bar with "Google Custom" and a search icon. Below that is a "Subscribe" section with options for RSS and e-mail. At the bottom right, there is a "Featured Articles" section with a link to "How I Taught My Mother Binary Numbers" and a sub-heading "Fathers, sons, daughters, brothers, sisters, aunts, and uncles should".

The **LOOP** Instruction

loop

loop label

loop

label

```
x      dd      1
sum    dd      0
```

```
      mov      ecx, 10
addUp: mov      eax, dword[x]
      add      dword[sum], eax
      inc      dword[x]
      loop     addUp           ;ecx←ecx-1
                                   ;if ecx!=0,
                                   ; goto addUp
```

loop

loop label

loop

label

```
x      dd      1
sum    dd      0
```

Label

```
addUp:  mov     ecx, 10
        mov     eax, dword[x]
        add    dword[sum], eax
        inc    dword[x]
        loop   addUp
```

**;ecx←ecx-1
;if ecx!=0,
; goto addUp**

loop

loop label

```
loop    label
;ecx ← ecx-1
;if ecx ≠ 0,
; goto label
;else continue
```

```
x      dd      1
sum    dd      0
```

Label

```
addUp: mov    ecx, 10
      mov    eax, dword[x]
      add   dword[sum], eax
      inc   dword[x]
```

```
loop  addUp ;ecx←ecx-1
      ;if ecx≠0,
      ; goto addUp
```


Labels

```
_start:      mov     eax, 4

              mov     ecx, 10

for1:        ...
              ...
              loop    for1

for2:        ...
              ...
              loop    for2
```

- Start with a **letter**
- End with a **colon** (when declared)
- Represent an **address** in the code section
- Must be **unique** in program

Tracing

One Example

eax
ecx

```
for:    mov     ecx, 3
        mov     eax, 1
        call   _printDec
        inc     eax
        loop   for                ;ecx←ecx-1
                                       ;if ecx!=0,
                                       ; goto for
```

eax

?

ecx

3

```
mov ecx, 3
```

```
mov eax, 1
```

```
for: call _printDec
```

```
inc eax
```

```
loop for
```

```
;ecx←ecx-1
```

```
;if ecx!=0,
```

```
; goto for
```

eax 1
ecx 3

```
    mov     ecx, 3  
    mov     eax, 1  
for:   call   _printDec  
       inc   eax  
       loop  for
```

```
;ecx←ecx-1  
;if ecx!=0,  
; goto for
```

1

eax

1

ecx

3

```
mov    ecx, 3
mov    eax, 1
for:   call  _printDec
      inc  eax
      loop for
```

```
;ecx←ecx-1
; if ecx!=0,
; goto for
```

1

eax

~~1~~ 2

ecx

3

```
mov    ecx, 3
mov    eax, 1
for:   call  _printDec
      inc  eax
      loop for
```

```
;ecx←ecx-1
; if ecx!=0,
; goto for
```

1

eax

~~1~~ 2

ecx

~~3~~ 2

```
    mov     ecx, 3
    mov     eax, 1
for:  call   _printDec
      inc   eax
      loop  for
```

```
;ecx←ecx-1
; if ecx!=0,
; goto for
```


12

eax

~~1~~ 2

ecx

~~3~~ 2

```
mov    ecx, 3
mov    eax, 1
for:   call  _printDec
      inc  eax
      loop for
```

```
;ecx←ecx-1
; if ecx!=0,
; goto for
```

12

eax

~~1~~/~~2~~3

ecx

~~3~~2

```
    mov     ecx, 3
    mov     eax, 1
for:  call   _printDec
      inc   eax
      loop  for
```

```
;ecx←ecx-1
; if ecx!=0,
; goto for
```

12

eax

~~1~~/~~2~~ 3

ecx

~~3~~/~~2~~ 1

```
    mov     ecx, 3
    mov     eax, 1
for:  call   _printDec
      inc   eax
      loop  for
```

```
;ecx←ecx-1
; if ecx!=0,
; goto for
```

123

eax

~~1~~/~~2~~3

ecx

~~3~~/~~2~~1

```
mov     ecx, 3
mov     eax, 1
for:    call  _printDec
        inc  eax
        loop for
```

```
;ecx←ecx-1
;if ecx!=0,
; goto for
```

123

eax

~~1~~/~~2~~/~~3~~ 4

ecx

~~3~~/~~2~~ 1

```
    mov     ecx, 3
    mov     eax, 1
for:  call   _printDec
      inc   eax
      loop  for
```

```
;ecx←ecx-1
; if ecx!=0,
; goto for
```

123

eax

~~1~~/~~2~~/~~3~~ 4

ecx

~~3~~/~~2~~/~~1~~ 0

```
mov     ecx, 3
mov     eax, 1
for:    call  _printDec
        inc  eax
        loop for
```

```
;ecx←ecx-1
; if ecx!=0,
; goto for
```

123

eax

~~1~~/~~2~~/~~3~~ 4

ecx

~~3~~/~~2~~/~~1~~ 0

```
mov    ecx, 3
mov    eax, 1
for:   call  _printDec
      inc  eax
      loop for
```

????

```
;ecx←ecx-1
;if ecx!=0,
; goto for
```

Example 1

Sum of 1..10


```
; computes sum(x, x+1, x+2, x+9)
```

```
; x = x+10
```

```
x      dd      1
```

```
sum    dd      0
```

```
      mov      ecx, 10
```

```
      mov      eax, dword[x]
```

```
addUP: add      dword[sum], eax
```

```
      inc      eax
```

```
      loop    addUp
```

```
;ecx←ecx-1
```

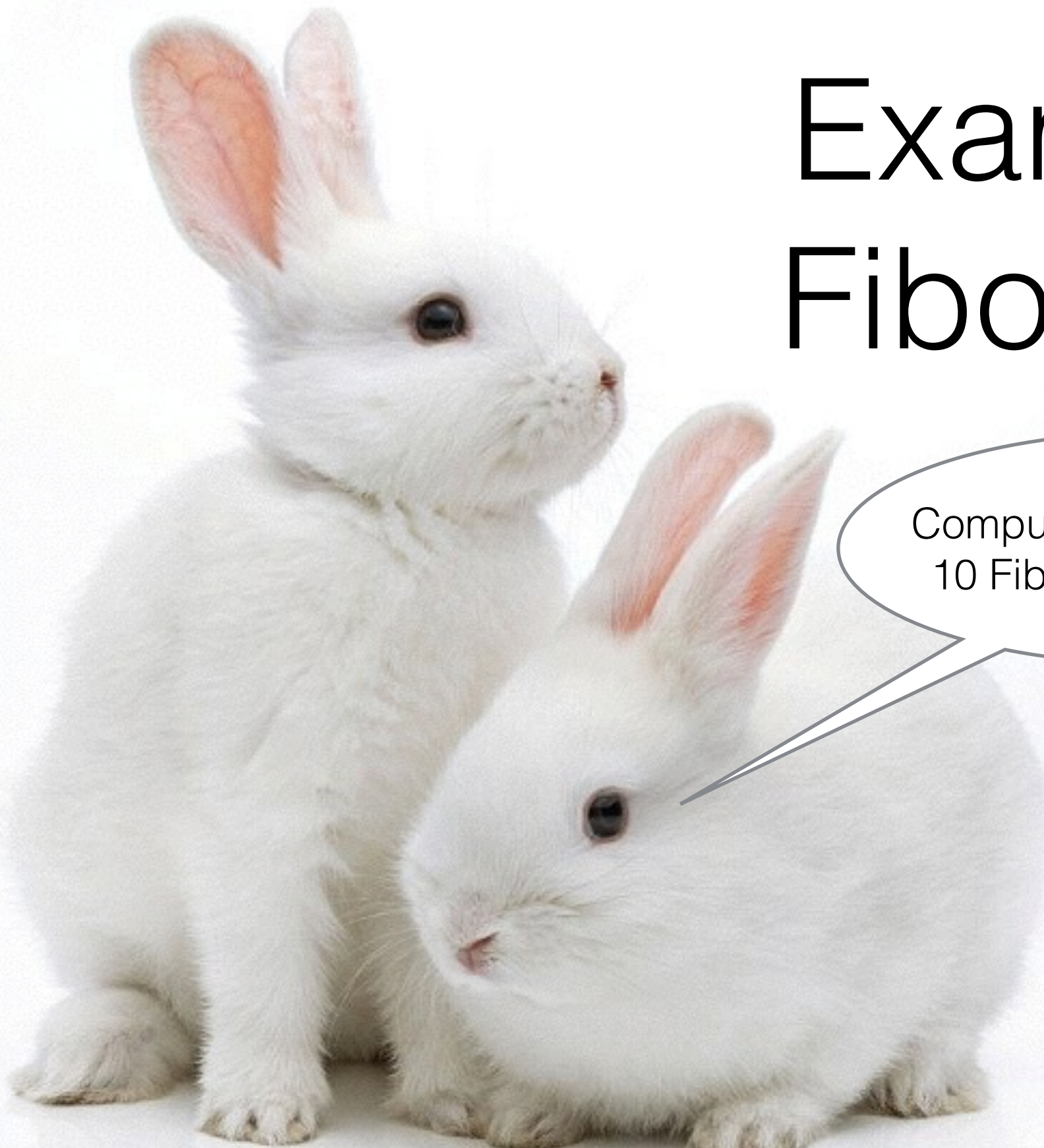
```
;if ecx!=0,
```

```
; goto addUp
```

```
      mov      dword[x], eax
```

Example 2

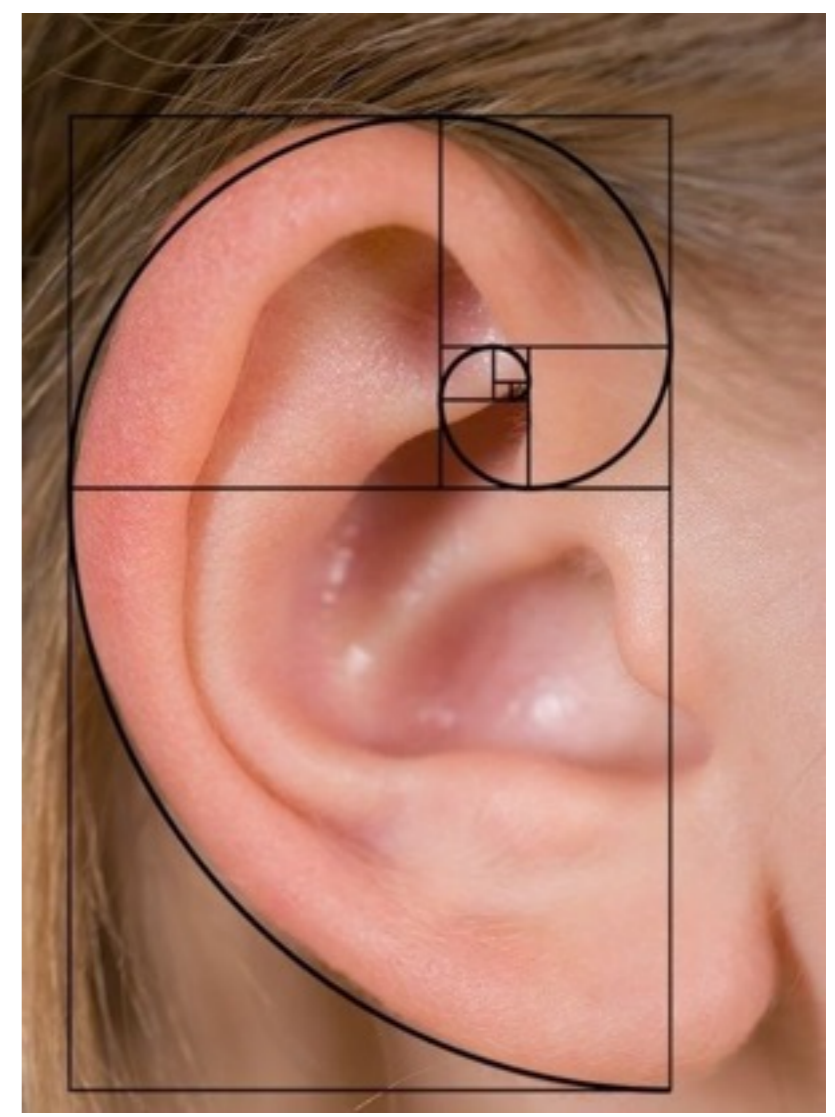
Fibonacci



Compute and print the first
10 Fib terms, 1, 2, 3, 5...



Examples of Fibonacci Sequences



First Step: Python

```
fibn = 1
fibn_1 = 1

print( fibn )

for i in range( 10-1 ):
    # pass down values
    fibn_2 = fibn_1
    fibn_1 = fibn

    # update fibn
    fibn = fibn_1 + fibn_2

    # print fibn
    print( fibn )
```

```
cs231a@aurora ~/ $ ./fibonacci.py
1
2
3
5
8
13
21
34
55
89
```

getcopy fibonacci.py

_start:

```
mov    eax, 1    ; eax is fibn
mov    ebx, 1    ; ebx is fibn-1
                    ; edx is fibn-2

call  _printDec
call  _println

mov    ecx, 10-1 ; we printed 1, 9 more to go
```

for:

```
mov    edx, ebx  ; fibn-2 <- fibn-1
mov    ebx, eax  ; fibn-1 <- fibn
add    eax, edx  ; fibn <- fibn-1 + fibn-2
call  _printDec
call  _println
loop  for
```

[getcopy fib.asm](#)