CSC231—Assembly

Week #9 — Spring 2017

Dominique Thiébaut
dthiebaut@smith.edu
2 Videos to Watch at a Later Time...

https://www.youtube.com/watch?v=FdMzngWchDk

https://www.youtube.com/watch?v=k2lZ1qsx4CM
https://www.youtube.com/watch?v=CgOcEZinQ2I
A 1-D Version

# # # # # # # # #
A 1-D Version
A 1-D Version
Rule 1: 0 neighbors
Rules of Life

Rule 1: 0 neighbors

Underpopulation
Rule 2: 1 neighbor
Rule 2: 1 neighbor

Right Environment
Rules of Life

Rule 3: 2 neighbors
Rule 3: 2 neighbors

Overpopulation
Problem of the Day(s): Implement 1D Game of Life in Assembly!
How to Approach This?
#Step 1: Write Algorithm in an More Comfortable Language...
# gameOfLife.py
# D. Thiebaut
# 1-Dimensional Game of Life

from __future__ import print_function
from __future__ import division
import random

def life( dish, N ):
    newGen = ""
    for i in range( 0, N ):
        neighbors = 0
        if i>0 and dish[i-1]!=' ': neighbors += 1
        if i < N-1 and dish[i+1]!=' ': neighbors += 1
        if neighbors == 1:
            newGen += "#"
        else:
            newGen += " "
    return newGen

def main():
    N = 40
    dish = (N//2-10)*"#" + 10*" #" + (N//2-10)*" "
    dish = dish[0:N]

    # print first generation
    print( dish )

    # repeat, for some generations
    for generation in range( 20 ):
        newGen = life( dish, N )
        print( newGen )
        dish = newGen

main()
```python
from __future__ import print_function
from __future__ import division

def life(dish, N):
    newGen = [0]*N
    for i in range(1, N-1):
        fate = dish[i-1] ^ dish[i+1] # ^ is xor
        newGen[i] = fate
    return newGen

def printDish(dish):
    print( ''.join( [ str(chr(ord(' ') + c)) for c in dish ] ) )

def main():
    N = 40
    dish = (N//2-10)*[1] + 10*[0,1] + (N//2-10)*[0]
    dish = dish[0:N]
    printDish(dish) # print first generation

    # repeat, for some number of generations
    for generation in range(20):
        newGen = life(dish, N)
        printDish(newGen)
        dish = newGen

main()
```

Game of Life
Python: V2

Same version but without tests

getcode GameOfLife_V2.py
Develop Assembly Program as a Class Exercise
We stopped here last time…
If-statements in Assembly
• **Jmp**: the jump instruction

• **flags** register

• **conditional** jumps (jne, je, jgt, jge, jlt, jle, ja, jb…)

Jumping around...

Start:

```
mov     ebx, Table          ;
jmp     there                ;

here:   mov     al, 1               ;
        mov     ecx, N              ;

there:  mov     byte[ebx+esi], al   ;
        inc     esi                  ;
        add     al, al                ;
        jmp     here                  ;
```
Jumping around...

_Start:

    mov     ebx, Table          
    jmp     there              

here:     mov     al, 1              
          mov     ecx, N         

there:    mov     byte[ebx+esi], al   
           inc     esi           
           add     al, al        
           jmp     here         

Jumping around...

_Start:

    mov    ebx, Table          ;
    jmp    there               ;

here:     mov    al, 1               ;
    mov    ecx, N              ;

there:   mov    byte[ebx+esi], al   ;
    inc    esi                 ;
    add    al, al              ;
    jmp    here               ;
Jumping around...

_Start:

```
mov     ebx, Table          ;
jmp     there               ;
```

_here:  mov     al, 1               ;
        mov     ecx, N              ;
        there:  mov     byte[ebx+esi], al   ;
                    inc     esi               ;
                    add     al, al              ;
                    jmp     here                ;
```
Jumping around...

_Start:
    mov    ebx, Table           ;
    jmp    there               ;

here:    mov     al, 1          ;
         mov     ecx, N          ;

there:   mov     byte[ebx+esi], al ;
         inc     esi            ;
         add     al, al            ;
         jmp     here            ;
Jumping around...

_Start:
    mov     ebx, Table          ;
    jmp     there              ;

here:    mov     al, 1           ;
         mov     ecx, N          ;
         there:   mov     byte[ebx+esi], al   ;
                  inc     esi         ;
                  add     al, al       ;
                  jmp     here        ;
Jumping around…

_Start:

    mov     ebx, Table          ;
    jmp     there             ;

here:   mov     al, 1               ;
        mov     ecx, N          ;

there:  mov     byte[ebx+esi], al  ;
        inc     esi           ;
        add     al, al         ;
    jmp     here             ;
Jumping around…

_Start:

```assembly
  mov      ebx, Table         ;
  jmp      there             ;

  here:   mov      al, 1       ;
          mov      ecx, N     ;

  there:  mov      byte[ebx+esi], al ;
          inc      esi      ;
          add      al, al    ;
          jmp      here     ;
```
jmp there ; "mov eip,there"
Flags Register
eax
ebx
cdx
edx
esi
edi

ADD
AND

ALU
### Examples

```assembly
_start:  nop
    nop                      ; immediate Flag values
    ; value AFTER the instruction

    mov    al, 0x43          ; 67
    sub    al, 0x43          ;                  PF ZF IF ID

    mov    al, 0x43          ; 67
    sub    al, 0x42          ; 66                  IF ID

    mov    al, 0x43          ; 67
    sub    al, 0x44          ; 68                  CF PF AF SF IF ID

    mov    al, 0x43          ; 67
    sub    al, 0xff          ; 255 or -1 CF PF AF IF ID

    mov    al, 0x43          ; 67
    sub    al, 0x81          ; 129 or -127 CF SF IF OF ID
```
Conditional Jumps
Example with Jnz

_Start:
    mov    ecx, 10

_for: . . .

    dec    ecx        ;ecx ← ecx - 1
    jnz    for        ;if previous op didn't result in 0 in ALU
                      ; then jump
                      ; else continue here...
Family of Conditional Jumps

- JE, JZ
- JNE, JNZ
- JG, JGE, JNL
- JL, JLE, JNG
Family of Conditional Jumps

EAX: 0xFFFF FFFF
EBX: 0x0000 0001

Which is greater?

- JE, JZ
- JNE, JNZ
- JG, JGE, JNL
- JL, JLE, JNG
Family of Conditional Jumps

EAX: 0xFFFF FFFF
EBX: 0x0000 0001

Which is greater?

- JE, JZ
- JNE, JNZ
- JG, JGE, JNL
- JL, JLE, JNG

- JE, JZ
- JNE, JNZ
- JA, JAE, JNB
- JB, JBE, JNA
How do we compare two quantities?
; if (a==b)
;     c = 3
; else
;     c = -1

    mov     eax, dword[a]        ;eax ← a
    mov     ebx, dword[b]        ;ebx ← b
    sub     eax, ebx             ;eax ← a - b, set flags
    jne     else                 ;if ZF flag not set, go to else

then:   mov      dword[c], 3          ;otherwise, a==b, set c to 3  
    jmp      done                 ;and skip else part

else:    mov      dword[c], -1         ;a!=b, set c to -1

done:    ...
; if (a == b)
;     c = 3
; else
;     c = -1

mov     eax, dword[a]    ; eax ← a
mov     ebx, dword[b]    ; ebx ← b
sub     eax, ebx         ; eax ← a - b, set flags
jne     else             ; if ZF flag not set, go to else

then:   mov     dword[c], 3  ; otherwise, a == b, set c to 3
        jmp     done        ; and skip else part

else:   mov     dword[c], -1  ; a! = b, set c to -1

done:   . . .
cmp is better, it subtracts eax from ebx, *but does not modify eax*

```assembly
; if (a==b)
;     c = 3
; else
;     c = -1
mov     eax, dword[a]        ;eax <— a
mov     ebx, dword[b]        ;ebx <— b
cmp     eax, ebx             ;eax <— a - b, set flags
jne     else                 ;if ZF flag not set, go to else
then:   mov     dword[c], 3          ;otherwise, a==b, set c to 3
        jmp     done                 ;and skip else part
else:   mov     dword[c], -1         ;a!=b, set c to -1
done:   . . .
```
Another example with cmp

; int a, c    // signed ints
; if (a < 10)
;   c = 3
; else
;   c = -1

mov     eax, dword[a]        ;eax ← a
cmp     eax, 10              ;eax ← a - 10, set flags
jnl     else                 ;if not less than 10, go to else
then:   mov     dword[c], 3          ;a<10, set c to 3
        jmp     done                 ;and skip else part
else:   mov     dword[c], -1         ;a >= 10, set c to -1
done:    . . .
Another example with cmp

```assembly
; int a, c   // signed ints
; if (a < 10)
;     c = 3
; else
;     c = -1

mov     eax, dword[a]        ;eax ← a
cmp     eax, 10              ;eax ← a - 10, set flags
j1      then                 ;if a<10 go to then

else:   mov     dword[c], -1         ;otherwise, a>=10, set c to -1
        jmp     done                 ;and skip then part

then:    mov     dword[c], 3          ;a < 10, set c to 3

done:   . . .
```
; int sum = 0
; for (int i=0; i<20; i+=2 ) {
;    sum += i;
; }

Translate this for-loop in assembly
Exercise 2

; unsigned int i, sum = 0
; for (i=0; i<4000000000; i+=2 ) {
;     sum += 1;
; }

Translate this for-loop in assembly.