파이썬 - HW4

임베디드스쿨1기

Lv1과정

2020. 08. 17

김인겸

디버거 사용법 정리

1. gcc test.c : test.c파일을 컴파일한다.

2. gdb a.out : test.c를 컴파일해서 만들어진 a.out 실행파일을gdb로 실행시킨다.

3. b main : main함수에 break point를 건다.

4. r : run(실행)

5. disas : 디스어셈블리어를 확인한다.

6. si : 디스어셈블리어 한줄을 실행시킨다.

7. l (listing) : c언어 레벨에서 현재 코드를 리스팅한다.

8. p (변수이름) : 변수에 들어있는 데이터를 확인한다.

9. p/x (변수이름) : 변수에 들어있는 데이터를 16진수로 확인한다.

디버거 사용법 정리

10. info reg : 현재 cpu의 레지스터 정보를 볼 수 있다.

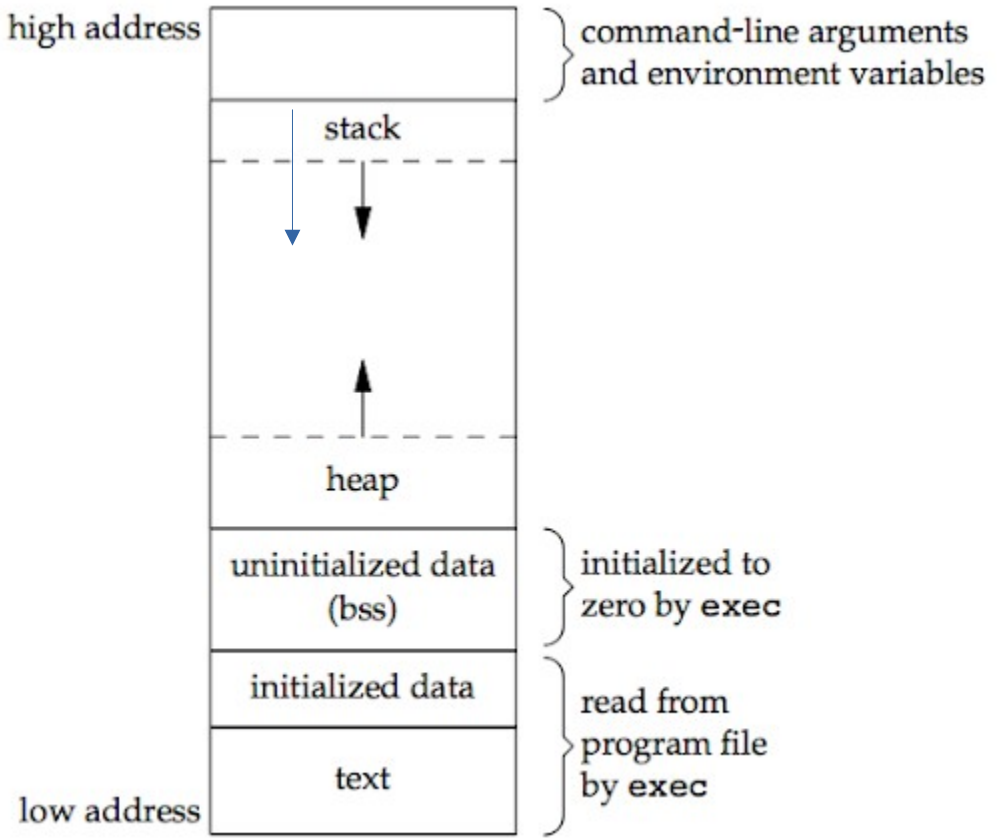
11. x(메모리주소) : 메모리 주소에 어떤 값이 들어있는지 확인할 수 있다.

12. c : 이미 한 번 run 시킨 상태에서 다음 break point 까지 실행한다.

메모리 레이아웃

1. stack은 메모리 영역 중의 하나이다.

2. stack에 할당된 값은 높은 주소값부터 낮은 주소값 순서대로 주소를 할당 받는다.
(little endian에서는 스택이 거꾸로 자란다)
주소값이 작아진다 → 메모리에 값이 쌓인다는 의미.



함수 호출 과 스택프레임

다음 소스코드의 어셈블리어를
살펴보자.

```c
#include <stdio.h>

int test(int num){
    return 3* num;
}

int main(void)
{
    int res;
    int num = 3;

    res = test(num);
    printf("res = %d\n", res);

    return 0;
}

```

소스코드

- 레지스터 종류

1. %rsp : 스택의 최상위 주소
   (최상위라 함은 낮은 주솟값을 의미)

2. %rbp : 스택의 베이스 주소(기준값을 의미)

3. %eax : 산술연산에 활용되는 레지스터.
   함수의 반환값이 eax에 저장됨!!
4. %edi : 임시저장 레지스터
5. %edx : eax와 같이 쓰이고 부호확장명령 에 쓰임
   (얘도 산술연산에 활용되는 레지스터)
6. %esi : 데이터를 조작하거나 복사시에 소스 데이터의
   주소를 저장하는 용도

함수 호출 과 스택프레임

main 함수 어셈블리어 살펴보기

```
0x000055555555515f <+0>:     endbr64
0x0000555555555163 <+4>:     push   %rbp
0x0000555555555164 <+5>:     mov    %rsp,%rbp
0x0000555555555167 <+8>:     sub    $0x10,%rsp
0x000055555555516b <+12>:    movl   $0x3,-0x8(%rbp)
0x0000555555555172 <+19>:    mov    -0x8(%rbp),%eax
0x0000555555555175 <+22>:    mov    %eax,%edi
0x0000555555555177 <+24>:    callq  0x555555555149 <test>
0x000055555555517c <+29>:    mov    %eax,-0x4(%rbp)
0x000055555555517f <+32>:    mov    -0x4(%rbp),%eax
0x0000555555555182 <+35>:    mov    %eax,%esi
0x0000555555555184 <+37>:    lea    0xe79(%rip),%rdi        # 0x5
0x000055555555518b <+44>:    mov    $0x0,%eax
0x0000555555555190 <+49>:    callq  0x555555555050 <printf@plt>
0x0000555555555195 <+54>:    mov    $0x0,%eax
0x000055555555519a <+59>:    leaveq
0x000055555555519b <+60>:    retq
```

함수 호출 과 스택프레임

1. push %rbp : rsp가 8byte만큼 감소하고 rbp(0x0)와 rsp사이의 공간에 rbp값을 넣는다(확인해보니 0x0)
이때 0x0은 main함수를 호출한 함수의 rbp값이다.
주솟값은 x64에서 8byte이므로 메모리에 8byte에 해당하는 rbp값이 생성된 것이다.

2. move %rsp,%rbp : rsp값을 rbp에 넣어준다 (rsp == rbp)

3. sub $0x10,%rsp : rsp에서 16byte뺀 값을 rsp에 넣어준다. rbp와 rsp사이에 main 함수의 임시 공간 16byte가 생겼다.

4. movl $0x3, -0x8(%rbp) : 숫자 3을 rbp에서 -8byte 된 주소값에 넣어준다.
(소스코드에서 res와 num이 int형이기 때문에 둘이 합치면 8byte이다,
res는 -0x4(%rbp)의 주솟값일 것이다)
이때 rbp값은 변하지않는다. rbp에서 8byte뺀 공간에 3을 넣어준 것이다.
(주소값이 높은 곳에서 낮은 곳으로 값이 저장되기 때문에 뺄셈을 하는 것이다)

5. mov -0x8(%rbp), %eax : rbp로부터 -8byte 떨어져있는 값(3) 을 eax레지스터에 저장한다

6. mov %eax, %edi : eax레지스터에 저장된 값 3을 edi 레지스터에 넣어준다
<여기까지가 함수를 호출하고 int num = 3; 까지 진행되는 과정이다.>

7. call 0x5555555555149<tset> : test함수를 호출.

함수 호출 과 스택프레임

test함수 호출과정 살펴보기

```
0x0000555555555149 <+0>:     endbr64
0x000055555555514d <+4>:     push   %rbp
0x000055555555514e <+5>:     mov    %rsp,%rbp
0x0000555555555151 <+8>:     mov    %edi,-0x4(%rbp)
0x0000555555555154 <+11>:    mov    -0x4(%rbp),%edx
0x0000555555555157 <+14>:    mov    %edx,%eax
0x0000555555555159 <+16>:    add    %eax,%eax
0x000055555555515b <+18>:    add    %edx,%eax
0x000055555555515d <+20>:    pop    %rbp
0x000055555555515e <+21>:    retq
```

0. test함수에 진입할 때 rsp 값이 8바이트 감소되었다. 이곳에 test함수가 끝난 다음의 주소값이 들어있다.

1. push &rbp : rsp는 8byte감소하고 이곳에 main함수의 rbp값이 저장된다.
2. mov %rsp, %rbp :
3. mov %edi, -0x4(%rbp) : main함수에서 저장한 edi값(3)을 rbp로부터 -4byte떨어진 주소에 넣어준다(int형이기 때문에 4byte)
4. mov -0x4(%rbp), %edx : rbp로부터 -4byte떨어진 주소에 있는 값(3)을 edx레지스터에 넣어준다.
5. mov %edx, %eax : edx값(3) 을 eax에 넣어준다.
6. add %eax, %eax : eax값(3)과 eax값(3)을 더해서 eax에 넣어준다(6)
7. add%edx, %eax : edx값(3)과 eax값(6)을 더해서 eax에 넣어준다(9) <여기까지가 3*num인듯>

8. pop %rbp : rbp값이 main함수의 rbp값으로 돌아간다.
9. retq : rsp의 값이 test함수를 호출하기 전의 값으로 돌아간다.

함수 호출 과 스택프레임

다시 main함수로 돌아와서

```
0x000055555555515f <+0>:     endbr64
0x0000555555555163 <+4>:     push   %rbp
0x0000555555555164 <+5>:     mov    %rsp,%rbp
0x0000555555555167 <+8>:     sub    $0x10,%rsp
0x000055555555516b <+12>:    movl   $0x3,-0x8(%rbp)
0x0000555555555172 <+19>:    mov    -0x8(%rbp),%eax
0x0000555555555175 <+22>:    mov    %eax,%edi
0x0000555555555177 <+24>:    callq  0x555555555149 <test>
0x000055555555517c <+29>:    mov    %eax,-0x4(%rbp)
0x000055555555517f <+32>:    mov    -0x4(%rbp),%eax
0x0000555555555182 <+35>:    mov    %eax,%esi
0x0000555555555184 <+37>:    lea    0xe79(%rip),%rdi        # 0x5
0x000055555555518b <+44>:    mov    $0x0,%eax
0x0000555555555190 <+49>:    callq  0x555555555050 <printf@plt>
0x0000555555555195 <+54>:    mov    $0x0,%eax
0x000055555555519a <+59>:    leaveq
0x000055555555519b <+60>:    retq
```

8. mov %eax, -0x4(%rbp) : eax값(9)(반환값)을 rbp에서 -4byte떨어진 주소값(int res)에 넣어준다.
9. ~ 13. 잘모르겠다 .print함수를 호출해서 문자열을 출력하는 과정이 아닐까 추측해봅니다.
10. leaveq : leaveq명령어 안에 pop rbp가 있다. Pop rbp를 함으로써 main함수를 호출한 함수의 rbp로 돌아간다.
11. retq : 함수끝.

함수 호출 과 스택프레임 결론

1. 함수를 호출하면, stack에 복귀 주소값과 호출된 함수 안에서 선언된 변수들이 메모리에 저장된다.
   함수호출이 끝나면, 함수 안에서 선언된 변수들이 모두 사라지고 리턴값만 eax레지스터에 저장된다.

2. 지역변수가 왜 함수 내에서만 사용되는지 알수 있었다.
   (함수호출이 끝나면 메모리에서 사라지기 때문에)
   추가로, 전역변수는 메모리의 data영역에 저장된다

3. 재귀함수 사용의 문제점이 스택프레임과 관련있다.
   재귀함수를 호출하면 적게는 수십, 많게는 수천개의 함수를 호출하게 되고 stack에 많은 값들이 생겼다가 지워졌다를 반복하여 메모리도 많이 쓰고 속도도 떨어지게 된다.

+ 함수를 호출하면 메모리에 경계선이 생긴다는 가르침의 의미
  : 기준점이 되는 rbp값을 경계선에 비유하신게 아닐까 생각해보았습니다~

예외처리 시 goto문 사용의 타당성

주파수가 3GHz 이고 20단계 파이프라인을 사용하는 CPU에 대해서 생각해보자.

if문으로만 예외처리를 할 때 추가적인 branch명령어가 100만개가 사용되었다고 하면
파이프라인이 100만번 깨지게 되고 20단계 파이프라인이라면 100만 * 20 클럭을 손해보게 된다.

3GHz에서 200만 클럭의 손해는 1/30^9 * 200만 = 0.0067초의 손실로 이어지지 않을까?..

따라서 for문이 여러번 중첩된 것을 예외처리 할 때에는 goto를 사용해 파이프라인이 깨지는 것을
막도록 하자!

문제풀이1. 주사위 맞추기

```
import random

number = random.randint(1,6)

user_num = (int)(input("주사위 숫자를 입력하세요 : "))

if user_num == number:
    print("승리")
else:
    print("실패! 주사위 숫자는 {} 였습니다". format(number))
```

```
주사위 숫자를 입력하세요 : 2
실패! 주사위 숫자는 7 였습니다
```

문제풀이2. 3슬롯머신

```python
import random

numbers = []
for i in range(0,3):
    numbers.append(random.randint(1,9))

if (numbers[0] == numbers[1] == numbers[2]):
    print(numbers, "\n승리!")
else:
    print(numbers, "\n실패")
```

```
[9, 9, 6]
실패
```

문제풀이3. 평균,중간값,분산,표준편차

```
import random
import math
SIZE = 200

numbers = []
for i in range(0,SIZE):
    numbers.append(random.randint(1,100))
print(numbers)

sum = 0          #평균 구하기
for i in numbers:
    sum += i
mean = sum / SIZE
print("평균은 {:f} 입니다". format(mean))

numbers.sort()    # 중간값 구하기
if (SIZE % 2) == 0:
    median = (numbers[(int)(SIZE/2)] + numbers[(int)((SIZE-1) / 2)] ) / 2
else:
    median = numbers[(SIZE - 1) / 2]
print("중간값은 {} 입니다.".format(median))
```

```
sum = 0         # 분산 구하기
for i in numbers:
    sum += (numbers[i] - mean) **2
variance = sum / SIZE
print("분산은 {} 입니다.".format(variance))

print("표준편차는 {} 입니다". format(math.sqrt(variance)))
```

```
[56, 29, 85, 87, 26, 73, 13, 3, 9, 96, 39, 47, 90, 10, 16, 43, 72, 81, 46, 100, 5, 77, 13, 50, 42, 97, 25, 7
7, 76, 88, 83, 94, 70, 48, 82, 20, 55, 68, 15, 34, 34, 95, 43, 21, 78, 66, 85, 71, 13, 4, 98, 83, 90, 45, 34,
23, 68, 70, 39, 68, 30, 1, 9, 32, 38, 74, 24, 86, 60, 37, 51, 76, 47, 91, 53, 34, 48, 53, 84, 37, 37, 12, 7,
22, 74, 16, 93, 71, 73, 19, 98, 28, 24, 93, 50, 35, 64, 23, 18, 63, 87, 12, 87, 38, 66, 2, 21, 48, 21, 84, 5
5, 96, 49, 93, 94, 52, 13, 26, 93, 45, 48, 81, 70, 64, 37, 90, 77, 76, 47, 26, 39, 91, 28, 63, 90, 28, 56, 1
2, 20, 64, 28, 51, 10, 96, 68, 47, 31, 100, 46, 45, 52, 73, 87, 98, 2, 74, 61, 35, 69, 82, 68, 96, 76, 18, 6
2, 51, 26, 42, 92, 14, 77, 23, 72, 7, 93, 98, 21, 79, 79, 14, 74, 62, 57, 2, 100, 65, 70, 31, 88, 44, 3, 25,
63, 13, 94, 92, 75, 71, 16, 35]
평균은 53.425000 입니다
중간값은 52.5 입니다.
분산은 762.3218750000021 입니다.
표준편차는 27.610177018628512 입니다
```

문제풀이4. 푸아송

```
import math

def poisson(y,x):
    return (pow(y,x) * math.exp(1/y) ) / math.factorial(x)

poisson_vlaue = poisson(10,1)

print("다음달 범죄가 발생할 확률은 {}% 입니다". format(poisson_vlaue))
```

```
다음달 범죄가 발생할 확률은 11.051709180756477% 입니다
```

문제풀이5. 스무고개

```
import random

answer = random.randint(0,1000000)

count = 20
for i in range(0,20):
    user_num = int(input("숫자를 입력하세요(남은기회:{}) : ".format(count)))
    if user_num == answer:
        print("정답입니다")
        break;
    elif(count != 1):
        if user_num > answer:
            print("DOWN")
        else:
            print("UP")
    else:
        print("실패했습니다 정답은 {} 입니다". format(answer))
    count -= 1
```

```
숫자를 입력하세요(남은기회:20) : 500000
UP
숫자를 입력하세요(남은기회:19) : 700000
UP
숫자를 입력하세요(남은기회:18) : 850000
DOWN
```

```
DOWN
숫자를 입력하세요(남은기회:6) : 737100
DOWN
숫자를 입력하세요(남은기회:5) : 737050
DOWN
숫자를 입력하세요(남은기회:4) : 737030
DOWN
숫자를 입력하세요(남은기회:3) : 737020
DOWN
숫자를 입력하세요(남은기회:2) : 737017
DOWN
숫자를 입력하세요(남은기회:1) : 737004
실패했습니다 정답은 735533 입니다
```

문제풀이6. 사다리게임(사다리를 만들 줄 모르겠습니다)

```python
import random


people = int(input("사람 수를 입력하세요 : "))
number_of_success = int(input("당첨의 갯수를 입력하세요 : "))

numberList = [number for number in range(1,people+1)]

answerList = random.sample(range(1,people+1),number_of_success) # 당첨숫자 생성

dict_answer = {}
for number in numberList:    #당첨숫자 딕셔너리 만들기
    for answer in answerList:
        if number == answer:
            dict_answer[number] = "!!!!!!!당첨!!!!!!!"
            break;
        else:
            dict_answer[number] = "꽝"

for k in dict_answer:
    print(k, ":", dict_answer[k])
```

```
사람 수를 입력하세요 : 20
당첨의 갯수를 입력하세요 : 2
1 : 꽝
2 : 꽝
3 : 꽝
4 : 꽝
5 : !!!!!!!!당첨!!!!!!!!
6 : 꽝
7 : 꽝
8 : 꽝
9 : 꽝
10 : 꽝
11 : 꽝
12 : 꽝
13 : 꽝
14 : 꽝
15 : 꽝
16 : 꽝
17 : !!!!!!!!당첨!!!!!!!!
18 : 꽝
19 : 꽝
20 : 꽝
```

문제풀이7

```
num = [1, 3]

for i in range(0,60):
    num.append(num[i]+num[i+1])

print(num)
print("{} 번째 숫자는 {} 입니다". format(23, num[23 - 1]))
```

```
[1, 3, 4, 7, 11, 18, 29, 47, 76, 123, 199, 322, 521, 843, 1364, 2207, 3571, 5778, 9349, 15127, 24476, 39603,
64079, 103682, 167761, 271443, 439204, 710647, 1149851, 1860498, 3010349, 4870847, 7881196, 12752043, 2063323
9, 33385282, 54018521, 87403803, 141422324, 228826127, 370248451, 599074578, 969323029, 1568397607, 253772063
6, 4106118243, 6643838879, 10749957122, 17393796001, 28143753123, 45537549124, 73681302247, 119218851371, 192
900153618, 312119004989, 505019158607, 817138163596, 1322157322203, 2139295485799, 3461452808002, 56007482938
01, 9062201101803]
23 번째 숫자는 64079 입니다
```

문제풀이8

```
num = [1, 1]
n = 57

for i in range(0,n):
    num.append(num[i]+num[i+1])

sum_odd = 0
sum_even = 0
for i in range(0,n):
    if (num[i] % 2) == 0:
        sum_even += num[i]
    else:
        sum_odd += num[i]

print("홀수 합 : {}". format(sum_odd))
print("짝수 합 : {}". format(sum_even))
print("홀수 합 - 짝수 합 : {}". format(sum_odd - sum_even))
```

```
홀수 합 : 478361013020
짝수 합 : 478361013020
홀수 합 - 짝수 합 : 0
```