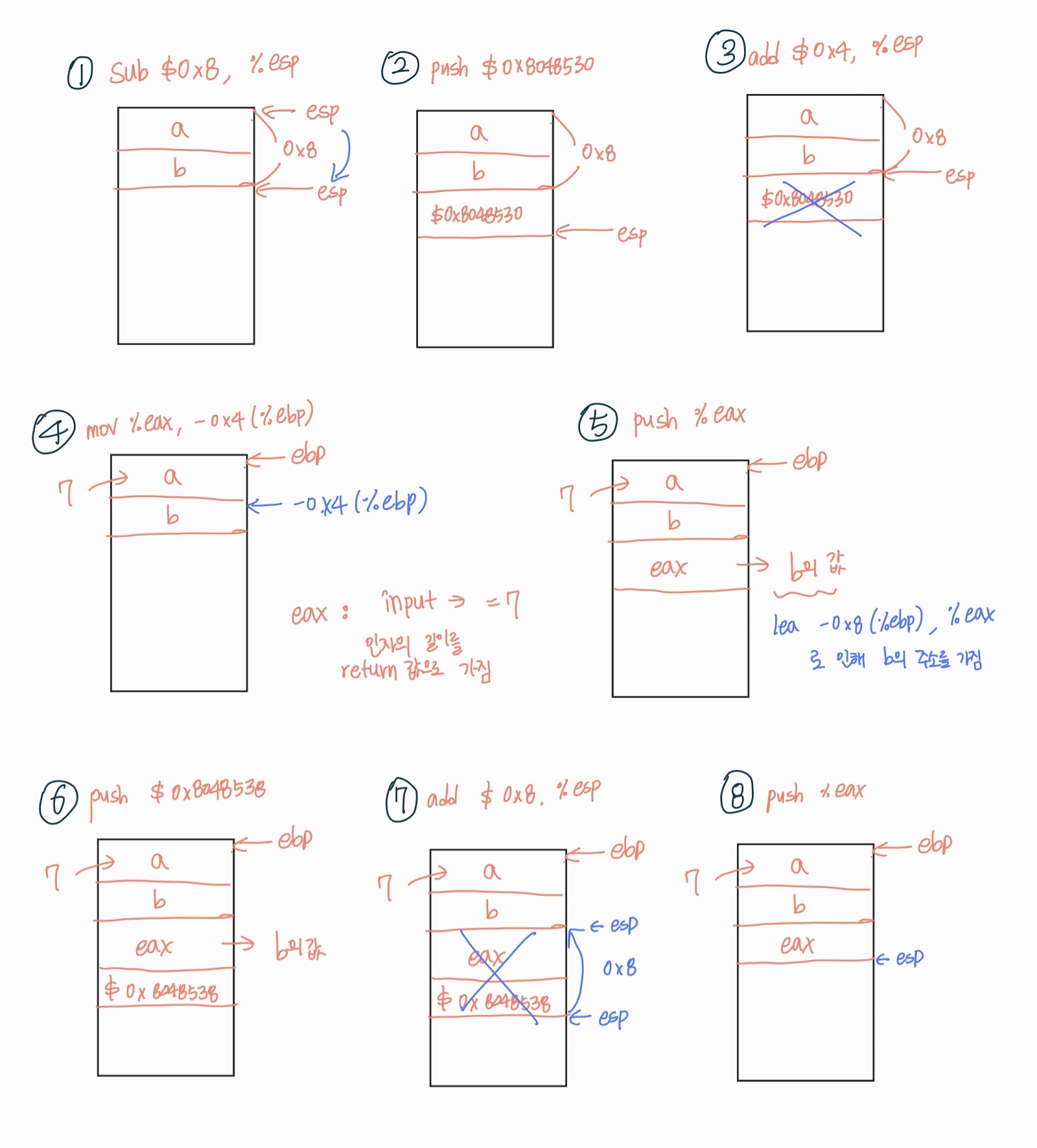
**1) prob1 Stack Frame 그리기**



텍스트이(가) 표시된 사진

자동 생성된 설명

**2) prob 1,2,3,4 풀기**

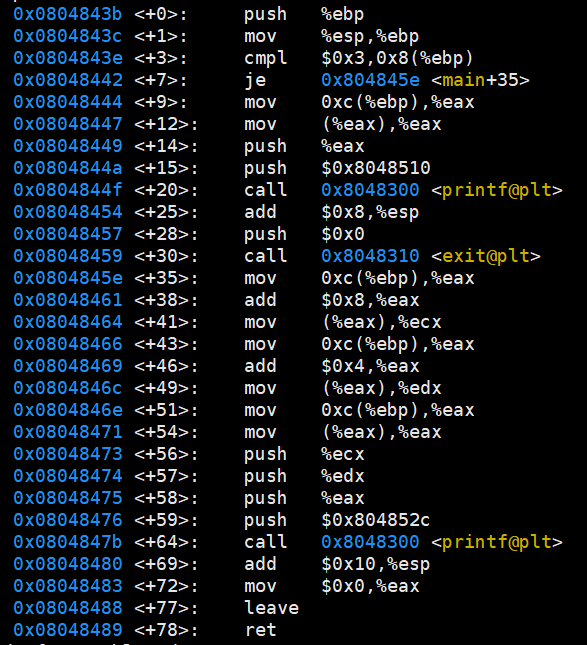
#1 prob1

텍스트이(가) 표시된 사진

자동 생성된 설명

|  |
| --- |
| #include<stdio.h> {  int main() {  int a;  int b;  a = printf("input->");  scanf("%d", &b);  printf("%d:%d", a, b);  } |

#2 prob2



|  |
| --- |
| #include<stdio.h>  int main(int argc, char\*\* argv) {    if (argc == 3) {  printf("%s [Parameter] [Parameter]\n", argv[0]);  exit(0);  }  else {  printf("%s %s %s\n", argv[0], argv[1], argv[2]);  }  } |

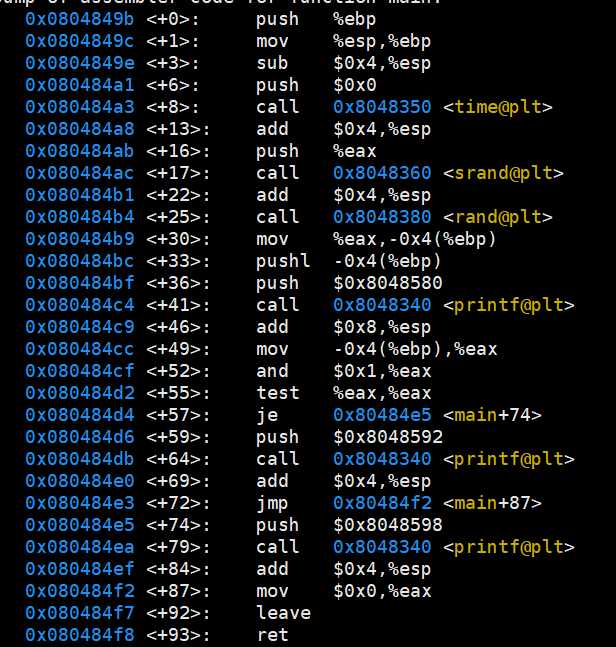
#prob3

텍스트이(가) 표시된 사진

자동 생성된 설명

|  |
| --- |
| #include<stdio.h>  int main() {  int a = 0;  int b;  for (b=0; b<100; b++)  a = a + b;  printf("%d\n", a);  } |

#prob4



|  |
| --- |
| #include<stdio.h>  #include<stdlib.h>  #include<time.h>  int main() {  int a;  srand(time(0));  a = rand();  printf("random value: %d\n", a);  if (a % 2 == 0) {  printf("Yes!");  }  else {  printf("nono~");  }  } |

#1 파일 디스크립터 조사 및 실습

**파일 디스크립터 :** 프로세스에서 특정 파일에 접근할 때 사용하는 추상적인 값

- 시스템으로부터 할당 받은 파일을 대표하는 0이 아닌 정수 값 (최대값 – OPEN\_MAX)

- 프로세스에서 열린 파일의 목록을 관리하는 테이블의 인덱스

+ 파일 – 유닉스 시스템에서 모든 것 (정규파일부터 디렉토리, 소켓, 파이브 등 모든 객체를 파일로 관리)

테이블이(가) 표시된 사진

자동 생성된 설명

프로세스가 실행 중에 파일을 열면 커널은 해당 프로세스의 파일 디스크립터 숫자 중 사용하지 않는 가장 작은 값을 할당함

→ 프로세스가 열려있는 파일에 시스템 콜을 이용해서 접근할 때, 파일 디스크립터 값을 이용해서 파일 지칭 가능

+ 시스템 콜 – 응용 프로그램의 요청에 따라 커널에 접근하기 위한 인터페이스



(C나 C++과 같은 고급 언어로 작성된 프로그램들은 직접 시스템 호출을 사용할 수 없기 때문에 고급 API를 통해 시스템 호출에 접근하게 하는 방법)



**#실습**

텍스트이(가) 표시된 사진

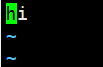
자동 생성된 설명

Stdin, stdout, stderr 에 각각 0,1,2가 할당됨을 확인 /

그 다음 숫자인 3,4가 할당됨을 확인하기 위해 txt 파일들을 생성 후 파일을 열어봄

텍스트이(가) 표시된 사진

자동 생성된 설명

 text.txt / test2.txt 파일 생성

#2 Calling Convention 조사 및 정리



함수 호출 규약 (Calling Convention) : 함수를 호출하는 방식에 대한 약속

|  |  |  |  |
| --- | --- | --- | --- |
| 규약 | 인자 전달 순서 | 인자 전달 매체 | Stack을 정리하는 함수 |
| **Cdecl** | ← | Stack | Caller |
| **Stdcall** | ← | Stack | Callee |
| **Fastcall** | ← | 레지스터 + Stack | Callee |

+ clrcall, thiscall, pascal, 등의 규약 존재

+ Caller / Callee란?

Caller – 콜러가 저장하는 레지스터, 값이 보존되지 않아도 되는 경우는 백업하지 않을 수 있음

Callee – 콜리가 저장하는 레지스터 , 콜러가 백업을 필요로하는 레지스터가 무엇인지 모르기에 무조건 백업해야함

**# \_\_CDECL 방식**

C / C++ 함수에서 기본적으로 사용되는 호출 규약

호출자가 스택을 정리

인자 순서 ← (오른쪽에서 왼쪽)

호출자가 피호출자를 호출시에 전달되는 인자의 개수를 알고 있음 – 가변 인수 함수를 만들 수 있음

+ 가변인수함수 : 인수의 개수와 타입이 미리 정해져 있지 않음. 그런 인수를 사용하는 함수

인자 전달 매체 – Stack을 사용

Stack Frame 정리 방법 – 함수가 호출한 Caller가 인자를 정리함

이름 수식 컨벤션 – 이름 앞에 \_가 붙음 (\_함수명)

**# \_\_Stdcall 방식**

Window API, Visual Basic에서 사용되는 표준 규약,

피호출자가 스택을 정리

인자 순서 ← (오른쪽에서 왼쪽)

호출을 당한 Callee가 함수를 종료하면서 인자를 정리함, 코드가 간결하지만 가변인자 사용 X

Stack Frame 정리 방법 – 호출을 당한 Callee가 함수를 종료하면서 인자를 정리함

이름 수식 컨벤션 - 이름 앞에 \_가 붙음 (\_함수명) 함수명 뒤에 인자 리스트의 바이트 크기를 @와 함께 표기함

(\_함수명@인자리스트크기)

**# \_\_Fastcall 방식**

Delphi 개발 툴의 기본 규약

인자 전달에 레지스터를 사용하므로 속도가 빠름

피호출자가 스택에서 인수들을 정리

인자 순서 ← (오른쪽에서 왼쪽)

Stack Frame 정리 방법 – 호출을 당한 Callee가 함수를 종료하면서 인자를 정리함.

이름 수식 컨벤션 - 이름 앞에 @가 붙음 (@함수명) 함수명 뒤에 인자 리스트의 바이트 크기를 @와 함께 표기함

(\_함수명@인자리스트크기)