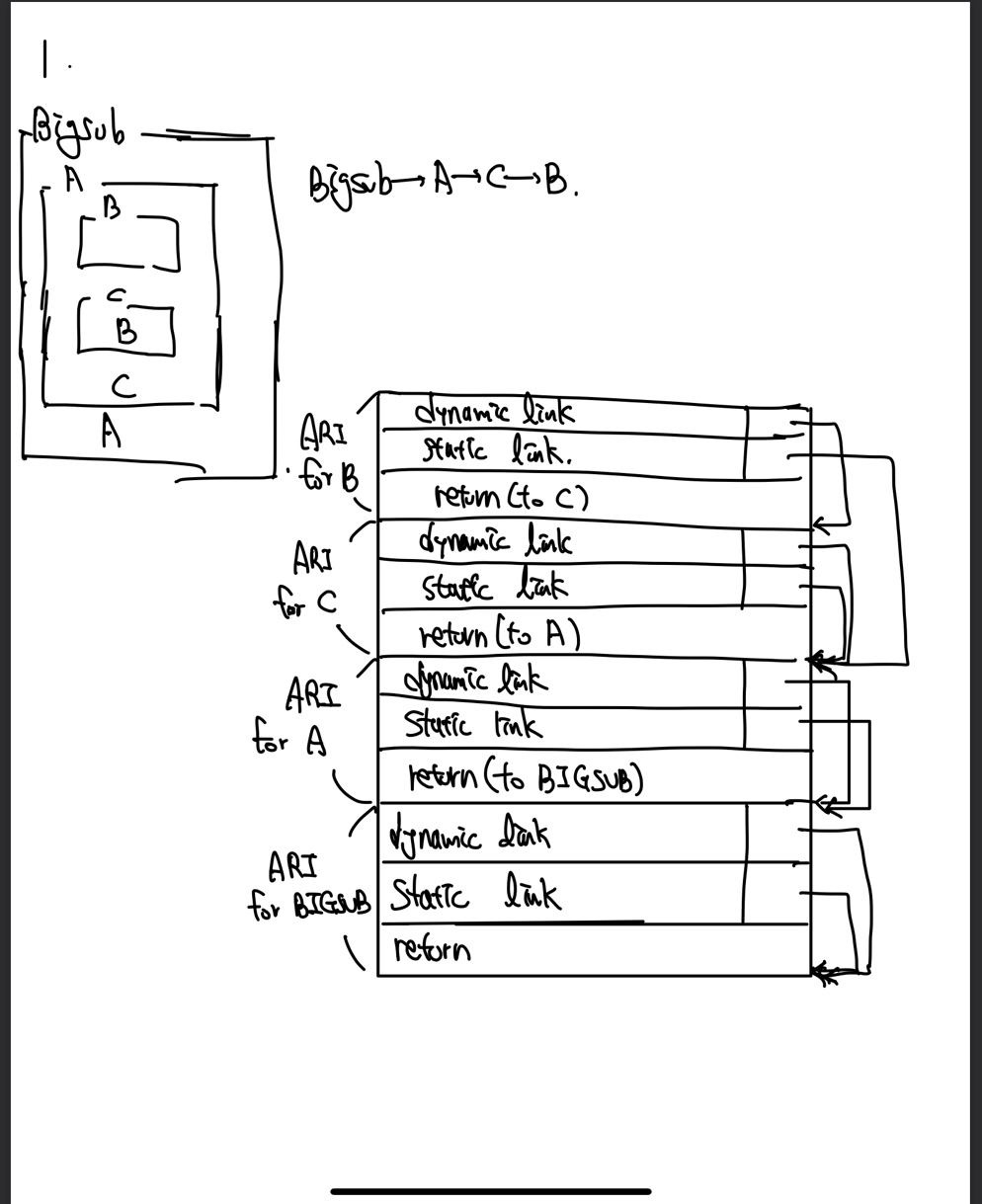
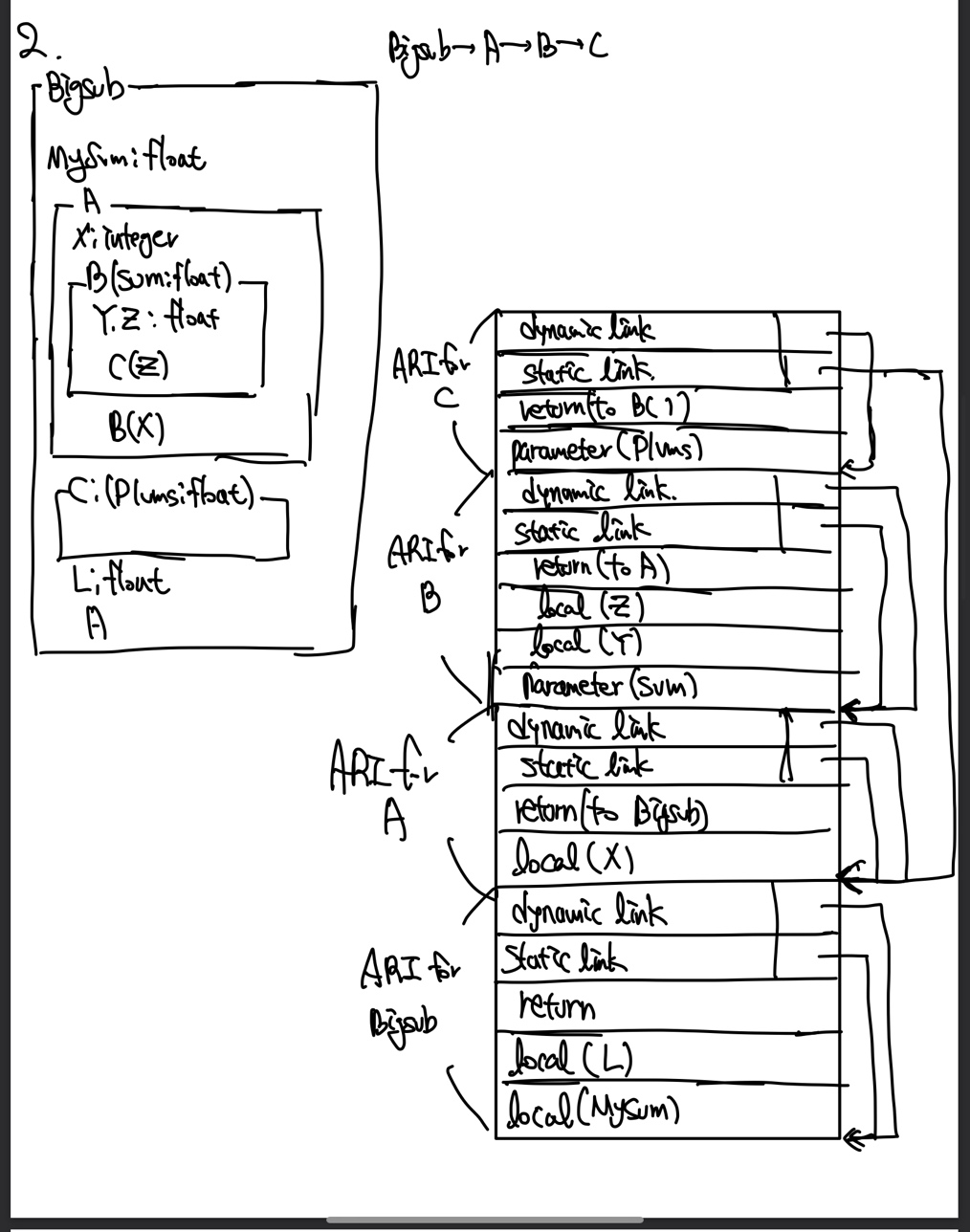
프로그래밍 언어 6차 과제

20181256 수학, 컴퓨터공학과 김도현

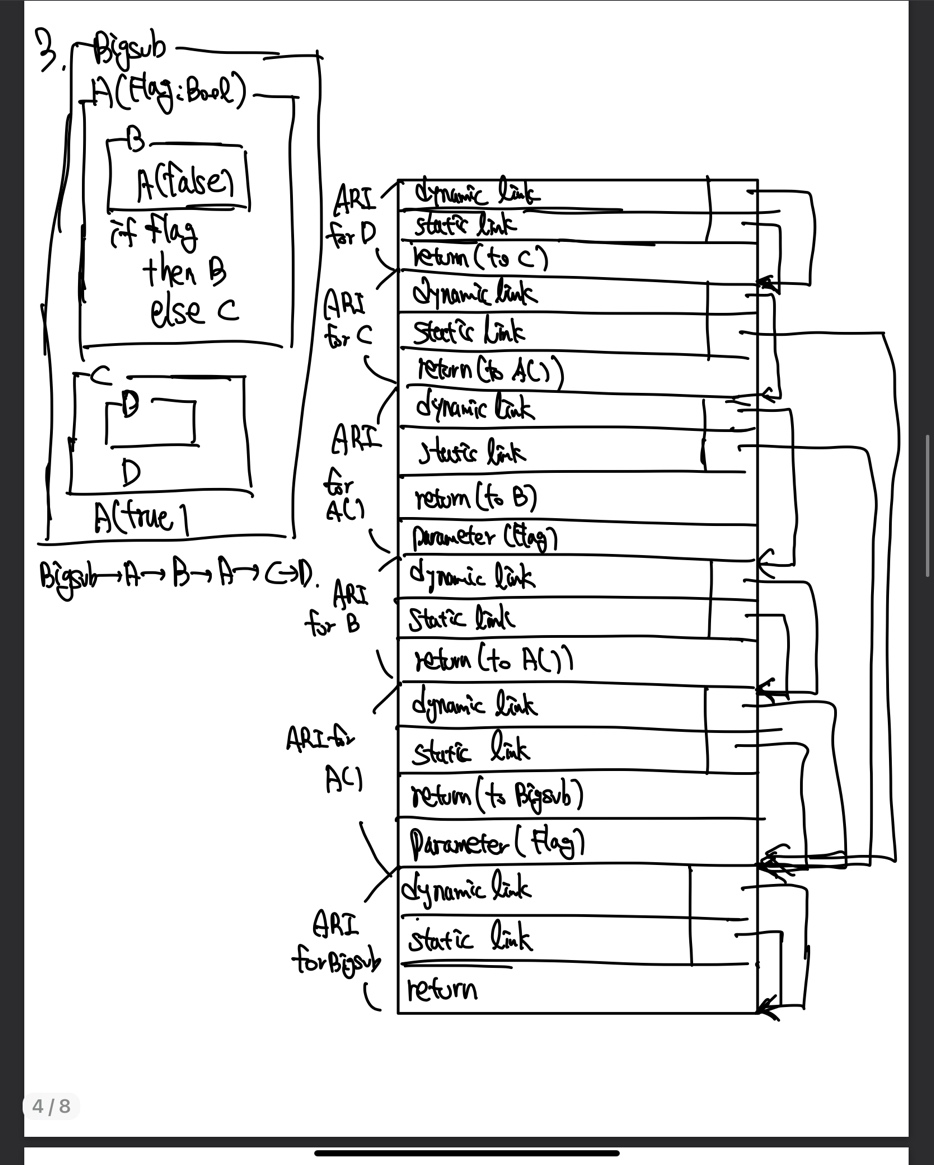
1-1.



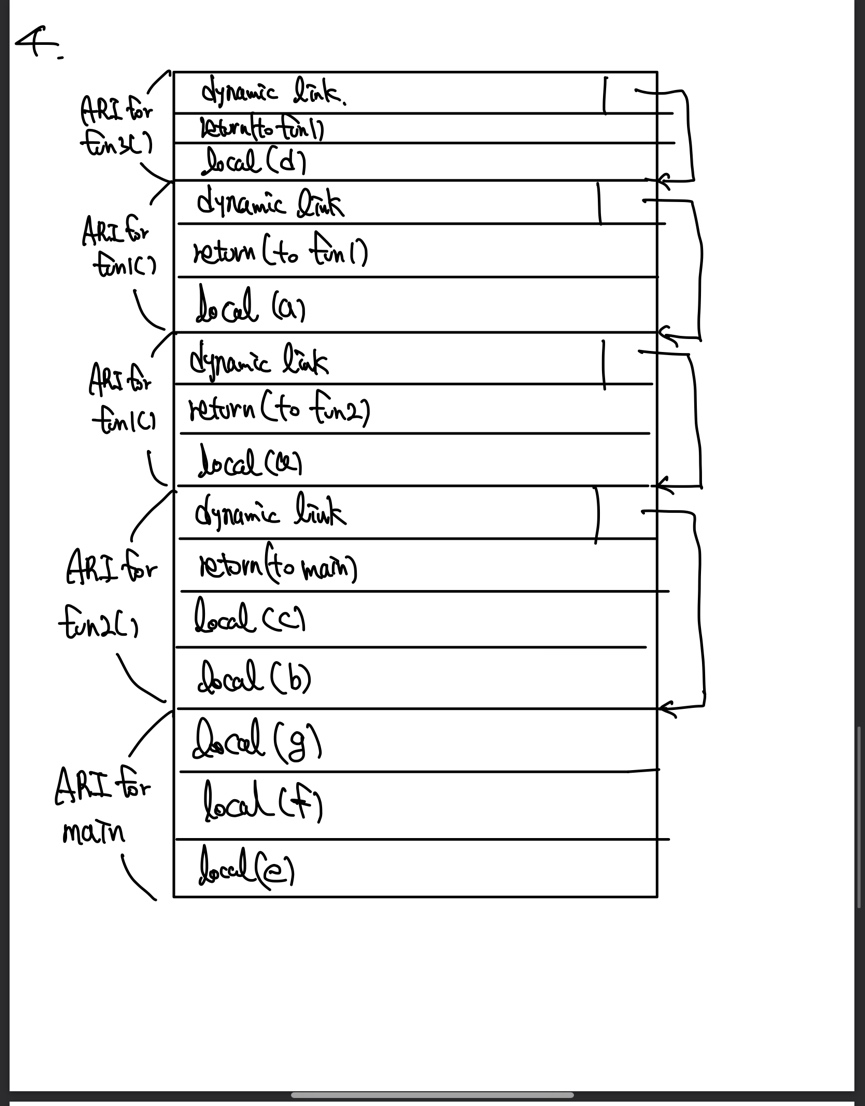
1-2.



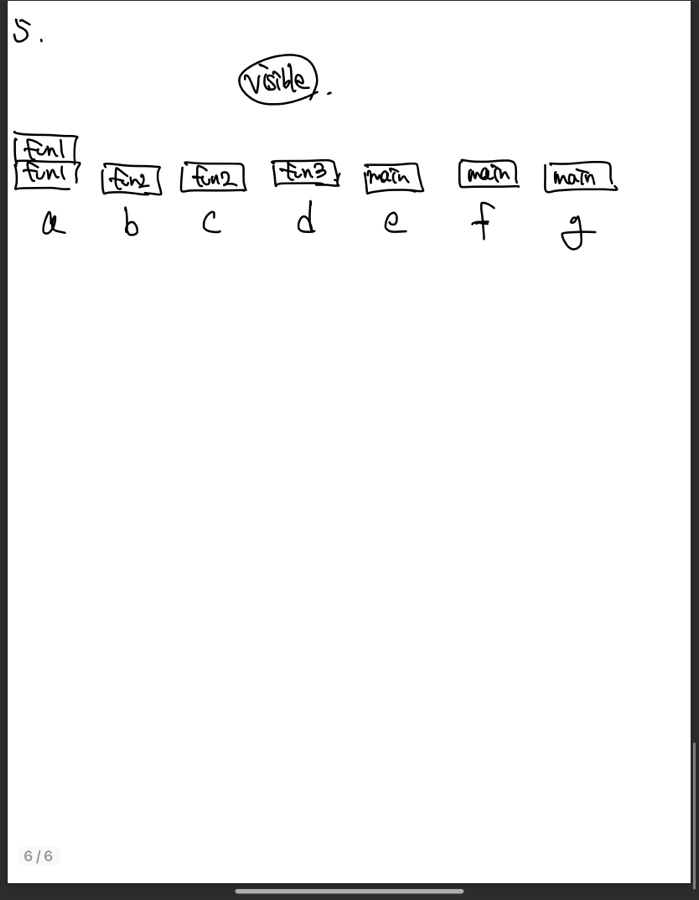
1-3.



1-4.



1-5.



1-6.

자바에서는 local variable 들이 static(정적)으로 선언 될 시 메모리에 고정적으로 할당되어 이전 값 유지가 가능하다.

2.

다음은 어셈블리 분석을 진행할 activation.c의 코드이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

어셈블리 코드를 분석할 activation.s 의 global및 main 영역이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

우선 main 함수에서 차례로 코드를 살펴보면 rbp에 rsp(스택의 현재 주소 스택의 top을 가르킴)를 주어 아무것도 없는 스택의 첫 시작 지점 주소를 둔다. 이후 rsp를 32만큼 감소 시켜 32byte를 확보한다. 데이터 보호를 위해 %fs:40 을 %rax에 넣고 %rax를 %rbp의 8 만큼 떨어진 곳에 넣는다. (canary)

Xorl eax eax 를 통해 산술 및 논리 연산을 수행하는 eax 레지스터를 초기화 시킨다.

Char bb[3] 1차원 배열의 경우 rbp(extened base pointer) 인 스택의 복귀 주소로부터 순차적으로 16,15,14 씩 떨어져 있음을 확인 할 수 있다. (local 변수의 activation record 위치)

이후 .LCO(%rip)를 %xmm0 으로 주어 Float 함수에 (3.2)라는 값을 보내는 float 타입의 actual parameter passing이 되도록 한다. Float함수 호출이 일어나고 %xmm0 로부터 받아온 값(return value)을 %eax에 넣고 이를 다시 rbp로 부터 24(float j) 떨어진 곳에 위치 시킨다.(activation record 위치) 함수 Integer 에 3을 넘기기 위해 숫자3을 edi에 넘겨(int 타입의 actual parameter passing) Integer함수 호출이 일어나 %eax에 저장된 return 받은 값(return value)을 rbp로 부터 20(int i)만큼 떨어진 곳에 위치한다.(return value의 activation record 위치) Rbp로부터 27이 떨어진 struct A의 aa 를 eax에 넣고 이를 다시 edi에 넘긴다.(struct 타입의 actual parameter passing) 이후 Structure 함수 호출이 일어나고 %al(%eax안의 1byte를 의미)에서 저장된 return 받아온 값을 rbp로부터 26만큼 떨어진 곳(bbb)에 위치하고 rax에 bb array 시작 주소를 넘기고 이 값을 rdi에 저장한다. Array같은 경우 rdi라는 메모리 이동이나 비교시에 목적지의 주소를 가리키는 포인터 레지스터를 사용했다. (array 타입의 actual parameter passing) 이후 Array 함수 호출이 일어나 %al(%eax안의 1byte를 의미)에서 저장된 return값(char 의 return value 전달 방식)을 rbp로부터 25만큼 떨어진 곳(ccc)에 위치시킨다. (return value의 activation record 위치)

텍스트이(가) 표시된 사진

자동 생성된 설명

우선 rbp 를 push 하고 스택의 현재 주소 스택의 top을 가르키는 rsp를 rbp에 주어 Integer의 activation record 시작 주소를 rbp(basepointer) 에 준다. %edi 로 부터 저장된 actual parameter값인 (3) 을 formal parameter로 받아 rbp로부터 4만큼 떨어진 곳에 위치시킨다.(parameter activation record 위치) 즉 여기서 integer에 관한 함수의 parameter passing 이 일어났다. 이후 다시 이 값을 eax 레지스터에 저장하고(변수 참조하기 위하여) 이 값을 다시 꺼내어 글로벌 변수 인 global\_var(%rip)에 저장시킨다. 여기서 global 변수를 어떻게 참조하는지 알 수 있다. 이후 글로벌 변수는 %edx 에 넣었다. 이후 다시 eax에 -4(%rbp) 즉 parameter(3)을 주고 addl 을 통해 %edx, %eax를 더해주고 해당 값은 %eax에 남아 있다. %eax 를 통해 main 함수에서 return value를 할 것이다. 그리고 %rbp를 pop 해주어 함수를 나간다.

텍스트이(가) 표시된 사진

자동 생성된 설명

우선 rbp 를 push 하고 스택의 현재 주소 스택의 top을 가르키는 rsp를 rbp에 주어 Float의 activation record 시작 주소를 rbp(basepointer) 에 준다. %xmm0 로 부터 저장된 actual parameter값인 (3.2) 을 formal parameter로 받아 rbp로부터 20만큼 떨어진 곳에 위치시킨다. (parameter의 activation record 위치) 즉 여기서 float에 관한 함수의 parameter passing 이 일어났다. Formal parameter를 사용하기 위해 %xmm0에 넣고 이 값을 local 변수인 temp에 준다. 이때 float temp local 변수를 rbp 로 부터 4만큼 떨어진 곳에 위치시키고(local 변수의 activation record 위치) 이를 다시 %xmm0(local temp를 사용하기 위해)에 넣는다. 그리고 %xmm1과 %xmm0 뺄셈을(temp-1) 시행한 후 이 결과값을 다시 rbp로부터 20 떨어진 곳에 넣는다.(formal parameter에 이 값을 준 것이다.) 그리고 이를 %xmm0에 다시 넣어주어 main 함수에서 return value를 받을 수 있게 하여 정상적으로 return 하고 함수를 종료 할 수 있게 해준다. 그리고 %rbp를 pop 해주어 함수를 나간다.

텍스트이(가) 표시된 사진

자동 생성된 설명

우선 rbp 를 push 하고 스택의 현재 주소 스택의 top을 가르키는 rsp를 rbp에 주어 Structure의 activation record 시작 주소를 rbp(basepointer) 에 준다. %edi의 8bit로 레지스터를 참조할 때 사용하는 %dil 로 부터 저장된 actual parameter값인 (struct A aa) 을 formal parameter로 rbp로부터 1만큼 떨어진 곳에 위치시킨다. (parameter의 activation record 위치) 즉 여기서 struct에 관한 함수의 parameter passing 이 일어났다. ‘e’ 를 뜻하는 $101 을 rbp로부터 1만큼 떨어진 곳(parameter 위치로 참조)에 넣고 이후 다시 이 값을 eax 레지스터에 저장한다. 해당 return 값은 %eax에 남아 있는데 main 함수에서 return value를 받을 때 사용하면 된다. 결국 char 형이다 보니 main에서는 %al 로 받으면 될 것이다. 그리고 %rbp를 pop 해주어 함수를 나간다.

텍스트이(가) 표시된 사진

자동 생성된 설명

우선 rbp 를 push 하고 스택의 현재 주소 스택의 top을 가르키는 rsp를 rbp에 주어 Integer의 activation record 시작 주소를 rbp(basepointer) 에 준다. %rdi 로 부터 저장된 actual parameter값인 (bb) 을 formal paramter로 받아 rbp로부터 24만큼 떨어진 곳에 위치시킨다. Array같은 경우 rdi라는 메모리 이동이나 비교시에 목적지의 주소를 가리키는 포인터 레지스터를 사용했다. (paramter의 activation record 위치) 즉 여기서 array에 관한 함수의 parameter passing 이 일어났다. 이 값은 %rax에 넣어 array parameter를 참조할 수 있게 한다. 2(%rax) 즉 bb[2](array 원소의 값 사용하는 방법) 를 임시 register %eax 에 넣고 rbp로 부터 1만큼 떨어진 곳에 char temp, local 변수를 위치시키는데(local변수의 activation record 위치) 이때 %eax의 1byte인 bb[2] 값을 부여한다. 다시 bb에 대한 연산을 하기 위해 %rax에 bb을 넣어주고(array parameter 변수 참조) leaq를 통해 %rdx에 bb[2] 의 주소를 준다.(array의 bb[2]의 값을 변경하기 위해 주소를 받는 레지스터) 그리고 1(%rax) 즉 bb[1] 값을 위에서 %eax에 준다. 이후 %al로 %eax 1byte만큼 읽어 %rdx에서 연산이 일어나 bb[2]의 값이 bb[1]의 값으로 바뀌었다. 그리고 bb[1] 주소 역시 leaq 를 통하여 %rdx에 준다. .(array의 bb[1]의 값을 변경하기 위해 주소를 받는 레지스터) 이후

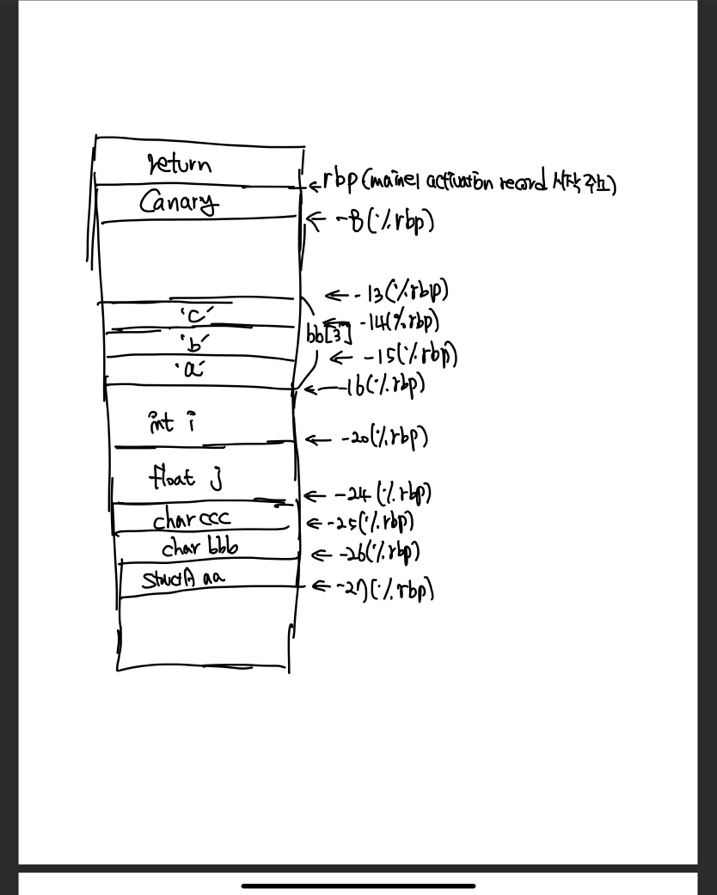
-1(%rbp) (local 변수의 값 사용법) 즉 temp 값을 %eax bb[1]에 준다. Char 형이라 1byte이기 때문에 %al로 읽어 %rdx 와 연산, 즉 bb[1] 값에 temp 값을 준다. (값이 바뀐다.) 바뀐 배열 을 다시 %rax에 준다.

해당 return 값은 1(%rax) 즉 bb[1] 인데 이는 %eax레지스터에 남아 있어 main 함수에서. Return value는 %eax를 사용하면 될 것이다. 결국 char 형이다 보니 caller에서는 %al 로 받으면 된다. 그리고 %rbp를 pop 해주어 함수를 나간다.

대략적인 main, Integer, Float 함수의 activation record 그림은 다음과 같다. Strcuture, Array 함수의 경우 비슷하기 때문에 생략하였다.

Main 함수에서의 activation record.

여기서 -13~27 까지 모두local 변수에 대한 값들이다.



ARI for Integer

텍스트이(가) 표시된 사진

자동 생성된 설명

ARI for Float

