# Cykor_1

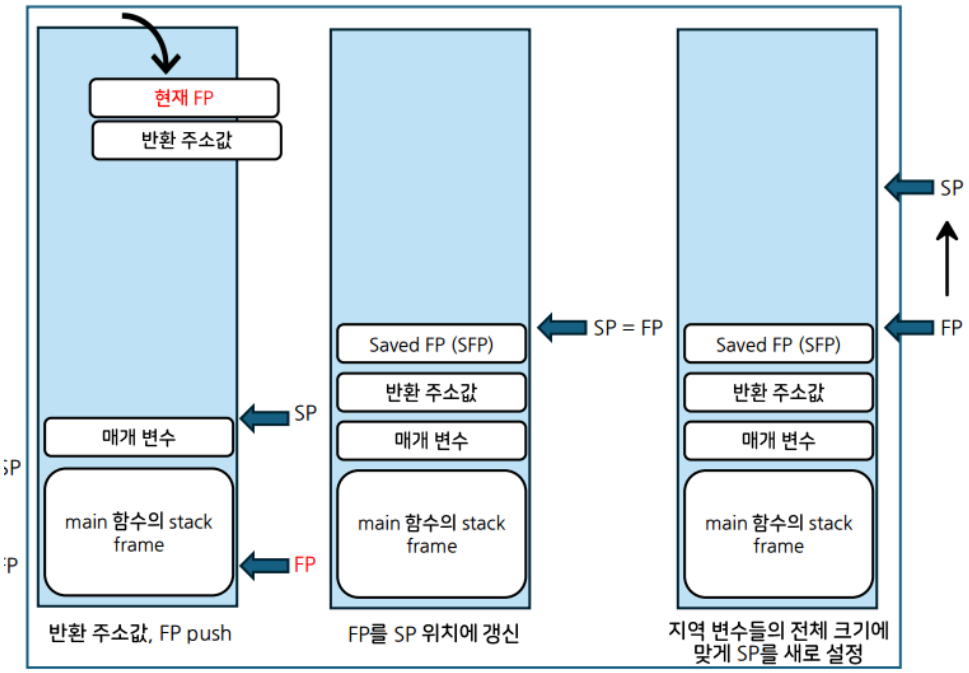
코드를 작성할 때 처음 부분은 주어진 코드를 그대로 사용하였습니다.

함수 4개를 추가로 작성하였는데 push, pop, function_prologue, function_epilogue 4개의 함수입니다.

push 함수부터 설명하면 오버플로우 방지를 위해 SP < STACK_SIZE - 1의 경우에만 일어나게 해두었고, SP + 1을 우선 해주고 값을 저장해주는데 이때 만약 num = -1이라면 Return Address가 저장이 됩니다. 이제 정보도 저장해주는데 이때 최대길이인 20자를 넘어가게 되면 19자까지만 저장하고 넘어갑니다. strncpy는 물론 위에서 길이를 지정해주었지만 '\0'을 만나면 종료하는 성질은  strcpy와 동일하게 가지고 있기 때문에 안전하게 `stack_info[SP][sizeof(stack_info[SP]) - 1] = '\0';` 를 작성하였습니다. 또한 이미 버퍼가 가득찬 경우 stack is full이 출력되게 하였습니다.

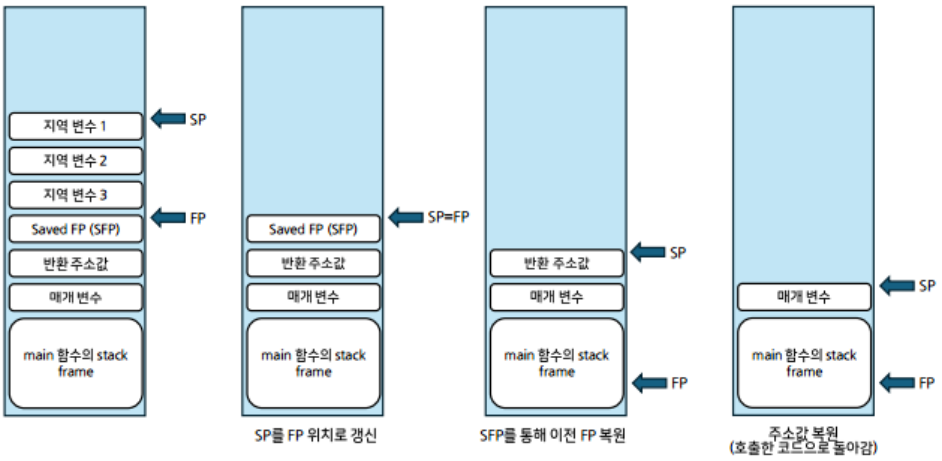
pop함수의 경우 빈 스택에 pop을 하면 안되므로 SP > -1인 경우에만 실행이 되도록 하였고, 현재 SP위치를 비우고 정보도 비웠습니다. 마지막으로 SP - 1을 하였습니다.

function_prologue의 경우 자료에 나와있는 그림대로




우선 매개변수는 직접 입력한다 하고(프롤로그 이전) 프롤로그 부분인 Return Address를 넣고 SFP를 넣어야 하므로 동일하게 진행하였습니다. 이제 지역변수의 개수에 맞추어서 SP를 올리고 공간을 마련해주었습니다.

function_epilogue의 경우에도 자료와 동일하게 진행하였는데




SP와 FP의 차이만큼 pop을 해줍니다(지역변수 제거). 이제 SFP, Return Address도 pop을 해주고 FP를 이전 FP로 다시 보내줍니다.

이제 함수들에서 매개변수의 경우 하나하나 push를 해주었고 지역변수의 경우 만들어놓은 공간에 들어가도록 하였고 함수 에필로그의 경우 콜러가 호출하도록 하였습니다. 에필로그 이후에는 콜리의 매개변수의 개수만큼 pop을 하여 제거하도록 하였습니다.