ATPCS(ARM-Thumb procedure call standard)

registers	용도	참고
r0-r3	arguments 및 return value(=r0)	현재 function은 이 register가 더 이상 사용되지 않는다면 이 register를 corrupt하여 scratch register로 사용할 수도 있음 (주: argument가 4개보다 많으면 stack을 사용함)
r4-r11	변수 저장용	현재 function은 이전 function에서 사용되던 이 register를 먼저 stack에 저장한 후 현재 function의 변수 저장용으로 사용함
r12	scratch register	현재 function은 이 register를 corrupt하여 scratch register로 사용할 수도 있음
r13(=sp)	stack pointer	full descending stack이고 8-byte align함
r14(=lr)	function return address를 가지는 link register	현재 function에서 다른 function을 call하면 이 register가 override되므로 이 register를 stack에 저장하여야 함
r15(=pc)	program counter	instruction 수행 시 PC에는 현재 instruction의 주소 + 8이 저장되어 있음

C to assembler 함수 호출

- C 코드에서 함수 호출
  - assembler 코드에 있는 함수 정의의 시작 주소에 있는 label이 함수 이름임
  - 일반 C 함수 호출과 같이 함수 호출함
- Assembler 코드에서 함수 정의
  - 함수 정의 시작 주소에 함수 이름으로 label 만듦
  - argument들(4개까지)은 r0-r3를 통하여 전달되니 이를 사용하여 프로그램 함
  - 함수 정의 내에서 ATPCS에 따라 register들을 사용함
  - return value는 함수 return 전에 r0에 저장해야함
  - 함수 return 시 "bx lr"을 사용

C to assembler 함수 호출 - 보기

/* C 코드 */

```
...
f4 = myfun4(a, b,
c, d);
```

/* Assembler 코드 */

```
.globl myfun4
myfun4:
  add r0, r0, r1
  add r0, r0, r2
  add r0, r0, r3
  bx lr
```

Assembler to C 함수 호출

- Assembler 코드에서 함수 호출
  - argument들(4개까지)은 "mov" 혹은 "ldr" 등의 instruction을 사용하여 r0-r1에 저장함
  - C 함수 이름을 label이라고 생각하고 "bl" instruction을 사용하여 C 함수 호출함
  - 함수 호출 후 return 값은 r0에 저장되니 이를 활용함
- C 코드에서 함수 정의
  - 일반 C 함수와 같이 정의함

Assembler to C 함수 호출 - 보기

```
/* Assembler 코드 */

   ...
   ldr r0, =format
   bl printf
   ...

format:
   .asciz "Hello!\n"
```

```
/* C 코드 */

int printf(const
char *format, ...)
{
   ...
}
```