2009년 06월 11일

컴파일 순서에 따라 원하는 컴파일을 해보자 -

이제까지 컴파일 공장에 관한 이야기를 했으니까, 이제 여러 가지 컴파일을 해보면 재미있을 것 같습니다만, 동의하시는지요? 후후. 컴파일 공장이야기에서 보면 object (*.o) file이 등장하게 되는데, 이 object file을 다시 한번 살펴보면 이렇게 해석할 수 있겠습니다.

어차피 컴파일 과정이란 C 컴파일러인 tcc 또는 armcc를 이용해서 ARM core에서 동작할 수 있는 Assembly를 만든 후, 다시 기계어 덩어리인 *.o file을 만들어 내게 되는데, 이 *.o file안에는 object file이라고 하여 ARM이 이해할 수 있는 기계어와 컴파일러가 최종 목적으로 하는 link단계를 처리할 수 있도록 symbol 정도 따위가 들어 있습니다. 따위라고 표현하니까 조금은 미안하네요. 결국엔 어떤 여러 개의 c file을 컴파일 하게 되면, 각각의 c file마다 o file이 결과물로 나오게 되어 있습니다. 뭐, 다들 아시는 얘기겠지만, 이런 원리라면 어떤 많은 양의 c file을 컴파일 한 후, 그 파일이 컴파일 되었는지를 알아 보려면 같은 이름의 .o file이 생성되었는지를 확인하면 간단하게 컴파일이 되었는지 확인할 수 있습니다. 예를 들어, spaghetti.c를 컴파일 하면, spaghetti.o가 생성되어 있어야 한다는 말이죠.

자, 그럼 이제 ARM compiler인 tcc를 이용해서 몇가지 컴파일을 해봐도 좋을 듯 합니다. 예제 하나를 만들어서 컴파일을 해보도록 하죠. 이제부터 사용할 요리 재료는 다음과 같습니다. 계속 사용할 것이니까, 기억해 두시는 것이 좋습니다.

spaghetti.h ----------

```
#define EQUAL =      /- 예제들에서 써먹을려고 일부러 EQUAL이라는 걸 두었다구요 *-
typedef struct {  bool memberBool;  int memberInt;  word memberWord;  }structure;
```

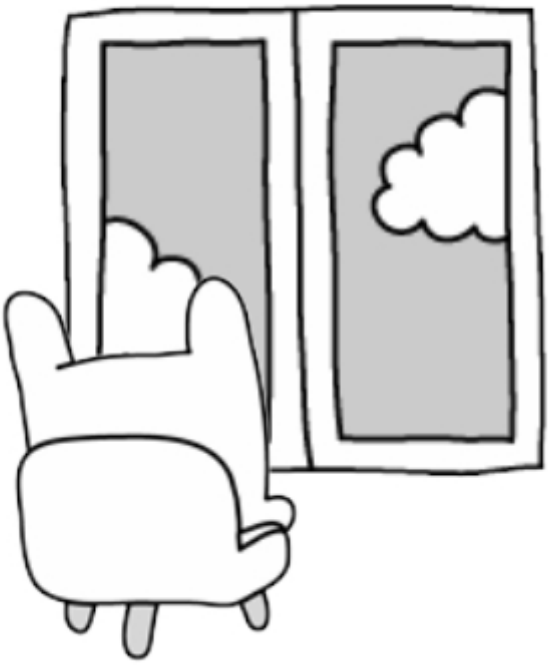
spaghetti.c ----------

```
#include "spaghetti.h"

int zi EQUAL 0; int rw EQUAL 3;
extern int relocate EQUAL 3;
extern structure recipes [3];
int add(int a, int b);

main () {  int stack;  volatile int local,local2,local3;    local EQUAL 3;  locall EQUAL 4;    local2 EQUAL add (local, local2);  stack +EQUAL local3;  return stack;  }
int add (int a, int b) {  return (a+b); }
```

컴파일을 연습할 file은 spaghetti.c이고, 이 file은 spaghetti.h를 포함합니다. spaghetti.c에는 보통 우리가 시작 entry로 사용하는 main()이 들어 있고, 그 안에서 sub routine function으로 add()라는 함수를 갖고 있습니다. 결국 main()이라는 함수에서 add()를 불러서 뭔가를 합하는 일을 하네요. 보통 이런 류의 글에서는 간단한 hello world를 출력하는 프로그램들을 만들지만 - 뭐, 저는 개인적으로 hello world 보다 뭔가를 더 하는 게 더 좋아요 - 이런 code는 variable들이, global, local 등, 메모리에 어떻게 자리 잡는지 보여주기 편하니까, 또는 계속 하나 가지고 울궈 먹을 수 있으니까, 이걸로 하겠습니다. - 라고 편하게 넘겨 버립니다 -



compile, Hardware, 레지스터, arm, AMBA, Assembly

Linked at at 2009/10/01 12:32

... ⓓ 원하는 컴파일을 해보자 ... more