시스템 소프트웨어

HW3

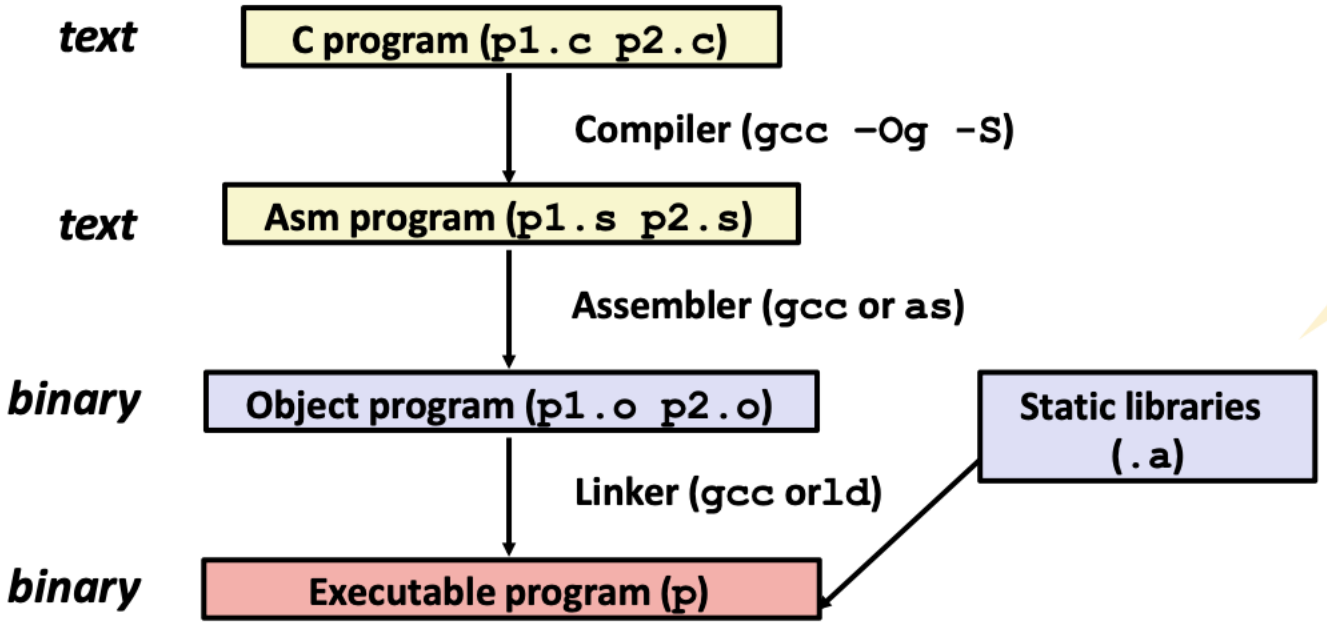
Moon Gi Seok

숙제 내용

- **다음 슬라이드에 나오는 5문제를 푸시오.**
  - 문제 1번을 제외한 나머지 2,3,4 는 실습 문제
    - 실습 서버를 이용해서 실습해보자.

- **제출 파일:**
  - 모든 내용을 통합해서 하나의 pdf 파일로 제출
    - 압축파일 및 기타 파일 업로드 금지

문제 1.



- 이 그림에서 1) 어셈블러(Assembler)의 역할, 2) 오브젝트 파일, 3) 정적 라이브러리가 무엇인지 짧게 설명하시오

문제 2.

- **오른쪽 코드를 vim을 통해서 직접 입력하시고, add.c 를 만들고, 다음의 세가지 캡쳐본을 만드시오.**
  - gcc –o 옵션을 통해 만들어진 실행 파일의 실행 결과
  - gcc –S 를 통해 만들어진 어셈블리
  - gcc –c 를 통해 만들어진 오브젝트 파일로부터 읽어들인 어셈블리어
    - 활용 키워드: objdump –d

```c
// add.c
#include <stdio.h>

int add(int a, int b) {
    return a + b;
}

int main() {
    int sum = add(3, 5);
    printf("Sum: %d\n", sum);
    return 0;
}
```

문제 3

- **gdb를 이용해보자. 코드는 이전 페이지와 동일**
  - gcc -g add.c -o add
    - add 실행 파일을 생성함
    - gcc -g 옵션의 의미
      - ✓ gcc -g를 사용하면 최적화를 수행하지 않고, 디버깅을 위한 정보를 바이너리에 추가하여 프로그램을 컴파일함.
    - 참고로 gcc –Og와 헷갈리지 말자
      - ✓ -Og 옵션은 최적화 옵션임
        - 디버깅과 최적화의 균형을 맞추기 위한 옵션 - 디버깅에 유용한 정보를 유지하면서도 코드의 성능을 향상시키기 위한 일부 최적화 작업을 수행함

```c
// add.c
#include <stdio.h>

int add(int a, int b) {
    return a + b;
}

int main() {
    int sum = add(3, 5);
    printf("Sum: %d\n", sum);
    return 0;
}
```

문제 3 Cont.

- gdb add 를 통해서 gdb를 실행하고, 아래 커멘드를 1) 수행한 결과와 2) 각 커멘드 동작 설명을 2-3줄 짧게 설명하시오.

```
(gdb) break main
(gdb) run
(gdb) list
(gdb) disassemble sumstore
(gdb) call add(1,7)
```

문제 4.

- **이전의 소스코드를 오른쪽과 같이 변경하였다.**
  - 이것에 대한 object 파일을 생성. 이를 objdump를 사용해 disassembly를 수행하면 결과는 다음 페이지와 같다.
  - 결과에서 함수를 호출하는 부분 callq 부분을 살펴보면 대상 함수 (add, minus, printf, printf 순) 의 주소가 0으로 채워져 있음을 볼 수 있다.
  - object 파일이 아닌, 실제 실행 파일을 활용하여 objdump를 수행하여 callq 뒤의 주소 부분이 어떻게 바껐는지를 실행했는지 캡쳐를 첨부해보자
    - 실행파일 기반 objdump 명령어 예시

      - ✓ 이 때, > a.out 부분은 화면에 내용을 출력하기에는 결과가 길어서, 해당 내용을 a.out 파일에 저장하려는 목적으로 기입되었음
    - 이를 기반으로 링커가 하는 일을 짧게 설명해보자
  - **문제 4번. 제출물 : call 부분 디스어셈블리 캡쳐 및 링커의 역할**

```c
#include <stdio.h>

int add(int a, int b){
    return a+b;
}

int minus(int a, int b){
    return a-b;
}

int main() {
    int sum = add(3, 5);
    int diff = minus(3, 5);
    printf("Sum: %d\n", sum);
    printf("Minus: %d\n", diff);
    return 0;
}
```

문제 4 (Cont.)

- 오브젝트 파일을 활용한 objdump disassembly 결과

```
Disassembly of section .text:

0000000000000000 <add>:
   0:   f3 0f 1e fa             endbr64
   4:   8d 04 37                lea    (%rdi,%rsi,1),%eax
   7:   c3                      retq

0000000000000008 <minus>:
   8:   f3 0f 1e fa             endbr64
   c:   89 f8                   mov    %edi,%eax
   e:   29 f0                   sub    %esi,%eax
  10:   c3                      retq

0000000000000011 <main>:
  11:   f3 0f 1e fa             endbr64
  15:   55                      push   %rbp
  16:   53                      push   %rbx
  17:   48 83 ec 08             sub    $0x8,%rsp
  1b:   be 05 00 00 00          mov    $0x5,%esi
  20:   bf 03 00 00 00          mov    $0x3,%edi
  25:   e8 00 00 00 00          callq  2a <main+0x19>
  2a:   89 c5                   mov    %eax,%ebp
  2c:   be 05 00 00 00          mov    $0x5,%esi
  31:   bf 03 00 00 00          mov    $0x3,%edi
  36:   e8 00 00 00 00          callq  3b <main+0x2a>
  3b:   89 c3                   mov    %eax,%ebx
  3d:   89 ea                   mov    %ebp,%edx
  3f:   48 8d 35 00 00 00 00    lea    0x0(%rip),%rsi        # 46 <main+0x35>
  46:   bf 01 00 00 00          mov    $0x1,%edi
  4b:   b8 00 00 00 00          mov    $0x0,%eax
  50:   e8 00 00 00 00          callq  55 <main+0x44>
  55:   89 da                   mov    %ebx,%edx
  57:   48 8d 35 00 00 00 00    lea    0x0(%rip),%rsi        # 5e <main+0x4d>
  5e:   bf 01 00 00 00          mov    $0x1,%edi
  63:   b8 00 00 00 00          mov    $0x0,%eax
  68:   e8 00 00 00 00          callq  6d <main+0x5c>
  6d:   b8 00 00 00 00          mov    $0x0,%eax
  72:   48 83 c4 08             add    $0x8,%rsp
  76:   5b                      pop    %rbx
  77:   5d                      pop    %rbp
  78:   c3                      retq
```