

# 컴파일러 최종과제

전자정보공학부

20160433

김민정

## 1. 실험방법

- 한 학기동안 신택스분석과 시멘틱분석이 포함된 컴파일러를 만들어보았다. 이번 과제에서는 code generator를 추가하여 원시코드를 어셈블리코드로 작성하는 컴파일러를 완성할 것이다.

- 원시 프로그램은 신택스 분석과 시멘틱 분석 과정을 통하여 신택스 트리와 관련 테이블들로 번역되었다. 이런 신택스 트리를 중간 언어의 형태로 번역하고 프로그램의 실행 효율을 높이기 위해 중간 언어프로그램을 최적화하는 과정을 하는 것이 바람직하나, 편의상 생략하고 신택스 트리로부터 바로 기계어 코드를 생성하고자 하였다. 시멘틱 분석과정을 통하여 모든 선언문들의 성격을 분석하고 코드 생성에 필요한 주소나 크기들이 계산되었으며 명령문들이나 수식이 올바르게 사용되고 있는지 분석하고 그 타입도 규칙에 따라 변환하여서 올바른 타입의 수식이나 명령들로 변환되었다. 그러므로 코드 생성기는 신택스 트리의 루트노드에 연결되어 있는 선언문 목록을 차례로 탐사하면서 다음과 같은 일을 수행한다.

\* 함수 선언문인 경우에는 활성화 레코드를 할당하는 코드를 생성하고, 함수의 몸체 즉 복합문을 탐사하여 그 안에 나타나는 모든 명령문에 대한 코드를 생성한다.

```
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$ ls
code_generator.c  lex.yy.c      print_syn.c  syntax.c     y.tab.c
kim.l            main.c        semantic.c   test_code    y.tab.h
kim.y            print_sem.c   simulator    type.h
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$ gcc -o a.exe code_generator.c lex.yy.c y.tab.c main.c semantic.c print_sem.c print_syn.c syntax.c
code_generator.c: In function 'gen_program':
code_generator.c:58:25: warning: passing argument 1 of 'gen_declaration_list' from incompatible pointer type [-Wincompatible-pointer-types]
    gen_declaration_list(node->clink);
                        ^
code_generator.c:15:6: note: expected 'A_ID * {aka struct s_id *}' but argument is of type 'struct s_node *'
    void gen_declaration_list (A_ID *);
    ^
code_generator.c: In function 'gen_expression':
code_generator.c:72:6: warning: assignment from incompatible pointer type [-Wincompatible-pointer-types]
    id = node->clink;
    ^
code_generator.c:98:24: warning: passing argument 3 of 'gen_code_i' makes integer from pointer without a cast [-Wint-conversion]
    gen_code_i(LITI, 0, id->init);
                    ^
code_generator.c:17:6: note: expected 'int' but argument is of type 'A_NODE * {aka struct s_node *}'
    void gen_code_i(OPCODE,int,int);
    ^
```

(생략)

```
syntax.c: In function 'setFunctionDeclaratorSpecifier':
syntax.c:307:43: warning: passing argument 1 of 'isNotSameFormalParameters' from incompatible pointer type [-Wincompatible-pointer-types]
    if (isNotSameFormalParameters(a->type, id->type->element_type))
                                    ^
syntax.c:44:9: note: expected 'A_ID * {aka struct s_id *}' but argument is of type 'A_TYPE * {aka struct s_type *}'
    BOOLEAN isNotSameFormalParameters(A_ID *, A_ID *);
    ^
syntax.c:307:52: warning: passing argument 2 of 'isNotSameFormalParameters' from incompatible pointer type [-Wincompatible-pointer-types]
    if (isNotSameFormalParameters(a->type, id->type->element_type))
                                    ^
syntax.c:44:9: note: expected 'A_ID * {aka struct s_id *}' but argument is of type 'struct s_type *'
    BOOLEAN isNotSameFormalParameters(A_ID *, A_ID *);
    ^
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$ ls
a.exe      kim.y      print_sem.c  simulator    type.h
code_generator.c  lex.yy.c  print_syn.c  syntax.c     y.tab.c
kim.l       main.c     semantic.c   test_code    y.tab.h
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$
```

- code generator에서는 초기화구문, switch문, structure와 관련된 부분은 만들지 않았다. 따라서 예제에서는 switch나 structure가 포함되어 있지 않는 코드들을 가지고 실험을 해볼 것이다.

- 주어진 어셈블리/인터프리터 프로그램을 실행하기 위해 다음과 같은 준비를 하였다. yacc을 이용하여 y.tab.c와 y.tab.h 생성한 후에(\$ yacc -d interp.y) lex를 이용하여 lex.yy.c 생성한다(\$ lex interp.l). 그 후 실행파일인 interp.exe를 만들었다. (\$ gcc -o interp.exe y.tab.c lex.yy.c interp.c lib.c) 이 실행파일을 이용하여 완성한 컴파일러를 통해 생성한 어셈블리 코드를 실행시켜 본다.

```
minjeong@minjeong-VirtualBox:~/mjeong/hw_final/simulator$ ls
interp.c interp.l interp.y lib.c type.h
minjeong@minjeong-VirtualBox:~/mjeong/hw_final/simulator$ yacc -d interp.y
minjeong@minjeong-VirtualBox:~/mjeong/hw_final/simulator$ ls
interp.c interp.l interp.y lib.c type.h y.tab.c y.tab.h
minjeong@minjeong-VirtualBox:~/mjeong/hw_final/simulator$ lex interp.l
minjeong@minjeong-VirtualBox:~/mjeong/hw_final/simulator$ ls
interp.c interp.l interp.y lex.yy.c lib.c type.h y.tab.c y.tab.h
minjeong@minjeong-VirtualBox:~/mjeong/hw_final/simulator$ gcc -o interp.exe y.tab.c lex.yy.c interp.c lib.c
y.tab.c: In function 'yyparse':
y.tab.c:1191:16: warning: implicit declaration of function 'yylex' [-Wimplicit-function-declaration]
    yychar = yylex ();
                ^
y.tab.c:1362:7: warning: implicit declaration of function 'yyerror' [-Wimplicit-function-declaration]
    yyerror (YY_("syntax error"));
            ^
interp.y: In function 'initialize':
interp.y:204:9: warning: assignment from incompatible pointer type [-Wincompatible-pointer-types]
    stack_f=stack;
            ^
interp.y:205:9: warning: assignment from incompatible pointer type [-Wincompatible-pointer-types]
    stack_c=stack;
            ^
```

(생략)

```
lib.c:106:5: warning: incompatible implicit declaration of built-in function 'strcpy'
lib.c:106:5: note: include '<string.h>' or provide a declaration of 'strcpy'
lib.c:111:12: warning: too many arguments for format [-Wformat-extra-args]
    printf("\%c",c);
            ^
minjeong@minjeong-VirtualBox:~/mjeong/hw_final/simulator$ ls
interp.c interp.l lex.yy.c type.h y.tab.h
interp.exe interp.y lib.c y.tab.c
minjeong@minjeong-VirtualBox:~/mjeong/hw_final/simulator$
```

## 2. 실험 내용

### 1) test1.c

이름과 나이, 키를 출력하는 프로그램이다.

```
test1.c (~/mjeong/hw_final) - VIM
1 void printAge(int age)
2 {
3     printf("age : %d\n", age);
4 }
5
6 void printHeight(float height)
7 {
8     printf("height : %f\n", height);
9 }
10
11 int main()
12 {
13     char *name;
14     int age;
15     float height;
16
17     name = "abc";
18     age=20;
19     height=159;
20
21     printf("name : %s\n", name);
22     printAge(age);
23     printHeight(height);
24
25     return 0;
26 }
27
28
29
```

● 컴파일러를 통하여 만들어진 어셈블러 프로그램은 다음과 같다.

```
a.asm (~/mjeong/hw_final) - VIM
1      INT    0, 72
2      SUP    0, main
3      RET    0, 0
4 printAge:
5      INT    0, 16
6      INT    0, 12
7      LDA    0, 12
8      LOD    1, 12
9      POP    0, 5
10     ADDR    0, printf
11     CAL    0, 0
12     RET    0, 0
13 printHeight:
14     INT    0, 16
15     INT    0, 12
16     LDA    0, 28
17     LOD    1, 12
18     POP    0, 5
19     ADDR    0, printf
20     CAL    0, 0
21     RET    0, 0
22 main:
23     INT    0, 24
24     LDA    1, 12
25     LDA    0, 44
26     STX    0, 0
27     POP    0, 1
28     LDA    1, 16
29     LITI    0, 20
30     STX    0, 0
31     POP    0, 1
32     LDA    1, 20
33     LOD    0, 52
34     STX    0, 0
35     POP    0, 1
36     INT    0, 12
"a.asm" 61L, 1076C
```

```

37      LDA    0, 56
38      LOD    1, 12
39      POP    0, 5
40      ADDR   0, printf
41      CAL    0, 0
42      INT    0, 12
43      LOD    1, 16
44      POP    0, 4
45      ADDR   0, printAge
46      CAL    0, 0
47      INT    0, 12
48      LOD    1, 20
49      POP    0, 4
50      ADDR   0, printHeight
51      CAL    0, 0
52      LDA    1, -4
53      LITI   0, 0
54      STO    0, 0
55      RET    0, 0
56 .literal    12  "age : %d\n"
57 .literal    28  "height : %f\n"
58 .literal    44  "abc"
59 .literal    52  159.000000
60 .literal    56  "name : %s\n"

```

- 어셈블리 프로그램을 시뮬레이터를 이용하여 실행하면 다음과 같은 결과가 나온다.

```

minjeong@minjeong-VirtualBox:~/mjeong/hw_final/simulator$ ./interp.exe a.asm
===== symbol =====
1: printf -1
2: malloc -2
3: scanf -3
4: main 19
5: printAge 3
6: printHeight 11
===== code =====
0: INT 0,72
1: SUP 0,19
2: RET 0,0
3: INT 0,16
4: INT 0,12
5: LDA 0,12
6: LOD 1,12
7: POP 0,5
8: ADDR 0,-1
9: CAL 0,0
10: RET 0,0
11: INT 0,16
12: INT 0,12
13: LDA 0,28
14: LOD 1,12
15: POP 0,5
16: ADDR 0,-1
17: CAL 0,0
18: RET 0,0
19: INT 0,24
20: LDA 1,12
21: LDA 0,44
22: STX 0,0

```

(생략)

```

47: CAL 0,0
48: LDA 1,-4
49: LITI 0,0
50: STO 0,0
51: RET 0,0

```

start execution

name : abc

age : 20

height : 159.000000

end execution

```
minjeong@minjeong-VirtualBox:~/mjeong/hw_final/simulator$
```



test1.c를 gcc로 실행하면 다음과 같은 결과가 나온다. 두 개가 동일한 결과라는 것을 확인할 수 있다.

```
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$ gcc -o gcc.exe test1.c
test1.c: In function 'printAge':
test1.c:3:2: warning: implicit declaration of function 'printf' [-Wimplicit-function-declaration]
   printf("age : %d\n", age);
   ^
test1.c:3:2: warning: incompatible implicit declaration of built-in function 'printf'
test1.c:3:2: note: include '<stdio.h>' or provide a declaration of 'printf'
test1.c: In function 'printHeight':
test1.c:8:2: warning: incompatible implicit declaration of built-in function 'printf'
   printf("height : %f\n", height);
   ^
test1.c:8:2: note: include '<stdio.h>' or provide a declaration of 'printf'
test1.c: In function 'main':
test1.c:21:2: warning: incompatible implicit declaration of built-in function 'printf'
   printf("name : %s\n", name);
   ^
test1.c:21:2: note: include '<stdio.h>' or provide a declaration of 'printf'
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$ ./gcc.exe
name : abc
age : 20
height : 159.000000
```

## 2) test2.c

```

test2.c (~/.mjeong/hw_final) - VIM
1 int main()
2 {
3     int money, m50000, m10000, m5000, m1000, m500, m100, m50, m10;
4
5     money = 278970;
6
7     m50000 = money / 50000;
8     printf("50000won => %d\n", m50000);
9     money = money - 50000 * m50000;
10
11     m10000 = money / 10000;
12     printf("10000won => %d\n", m10000);
13     money = money - 10000 * m10000;
14
15     m5000 = money / 5000;
16     printf("5000won => %d\n", m5000);
17     money = money - 5000 * m5000;
18
19     m1000 = money / 1000;
20     printf("1000won => %d\n", m1000);
21     money = money - 1000 * m1000;
22
23     m500 = money / 500;
24     printf("500won => %d\n", m500);
25     money = money - 500 * m500;
26
27     m100 = money / 100;
28     printf("100won => %d\n", m100);
29     money = money - 100 * m100;
30
31     m50 = money / 50;
32     printf("50won => %d\n", m50);
33     money = money - 50 * m50;
34
35     m10 = money / 10;
36     printf("10won => %d\n", m10);
37     money = money - 10 * m10;
38
39     return 0;
40 }

```

- 컴파일러를 통하여 만들어진 어셈블러 프로그램은 다음과 같다.

```

a.asm (~/.mjeong/hw_final) - VIM
1     INT     0, 164
2     SUP     0, main
3     RET     0, 0
4 main:
5     INT     0, 48
6     LDA     1, 12
7     LITI    0, 278970
8     STX     0, 0
9     POP     0, 1
10    LDA     1, 16
11    LOD     1, 12
12    LITI    0, 50000
13    DIVI    0, 0
14    STX     0, 0
15    POP     0, 1
16    INT     0, 12
17    LDA     0, 12
18    LOD     1, 16
19    POP     0, 5
20    ADDR    0, printf
21    CAL     0, 0
22    LDA     1, 12
23    LOD     1, 12
24    LITI    0, 50000
25    LOD     1, 16

```

(생략)

```

166    MULI    0, 0
167    SUBI    0, 0
168    STX     0, 0
169    POP     0, 1
170    LDA     1, -4
171    LITI    0, 0
172    STO     0, 0
173    RET     0, 0
174 .literal   12    "50000won => %d\n"
175 .literal   32    "10000won => %d\n"
176 .literal   52    "5000won => %d\n"
177 .literal   72    "1000won => %d\n"
178 .literal   92    "500won => %d\n"
179 .literal  112    "100won => %d\n"
180 .literal  132    "50won => %d\n"
181 .literal  148    "10won => %d\n"

```

- 어셈블러 프로그램을 시뮬레이터를 이용하여 실행하면 다음과 같은 결과가 나온다.

```
minjeong@minjeong-VirtualBox:~/mjeong/hw_final/simulator$ ./interp.exe a.asm
===== symbol =====
 1: printf -1
 2: malloc -2
 3: scanf -3
 4: main 3
===== code =====
 0: INT 0,164
 1: SUP 0,3
 2: RET 0,0
 3: INT 0,48
 4: LDA 1,12
 5: LITI 0,278970
 6: STX 0,0
 7: POP 0,1
 8: LDA 1,16
 9: LOD 1,12
10: LITI 0,50000
11: DIVI 0,0
12: STX 0,0
13: POP 0,1
14: INT 0,12
```

(생략)

```
167: POP 0,1
168: LDA 1,-4
169: LITI 0,0
170: STO 0,0
171: RET 0,0
start execution
50000won => 5
10000won => 2
5000won => 1
1000won => 3
500won => 1
100won => 4
50won => 1
10won => 2
end execution
minjeong@minjeong-VirtualBox:~/mjeong/hw_final/simulator$
```

test2.c를 gcc로 실행하면 다음과 같은 결과가 나온다. 두 개가 동일한 결과라는 것을 확인할 수 있다.

```
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$ gcc -o gcc.exe test2.c
test2.c: In function 'main':
test2.c:8:2: warning: implicit declaration of function 'printf' [-Wimplicit-function-declaration]
   printf("50000won => %d\n", m50000);
   ^
test2.c:8:2: warning: incompatible implicit declaration of built-in function 'printf'
test2.c:8:2: note: include '<stdio.h>' or provide a declaration of 'printf'
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$ ./gcc.exe
50000won => 5
10000won => 2
5000won => 1
1000won => 3
500won => 1
100won => 4
50won => 1
10won => 2
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$
```



### 3) test3.c

```
test3.c (~/mjeong/hw_final) - VIM
1 int main() {
2
3     int sec, s, hour, minute, second;
4
5     sec = 54321;
6     s = sec;
7
8     hour = sec / (60 * 60);
9     sec = sec - hour * 60 * 60;
10
11     minute = sec / 60;
12     sec = sec - minute * 60;
13
14     second = sec;
15
16     printf("%dsec is %02d:%02d:%02d.\n", s, hour, minute, second);
17
18     return 0;
19 }
```

- 컴파일러를 통하여 만들어진 어셈블리 프로그램은 다음과 같다.

```
a.asm (~/mjeong/hw_final) - VIM
1     INT    0, 44
2     SUP    0, main
3     RET    0, 0
4 main:
5     INT    0, 32
6     LDA    1, 12
7     LITI   0, 54321
8     STX    0, 0
9     POP    0, 1
10    LDA    1, 16
11    LOD    1, 12
12    STX    0, 0
13    POP    0, 1
14    LDA    1, 20
15    LOD    1, 12
16    LITI   0, 60
17    LITI   0, 60
18    MULI   0, 0
19    DIVI   0, 0
20    STX    0, 0
21    POP    0, 1
22    LDA    1, 12
23    LOD    1, 12
24    LOD    1, 20
25    LITI   0, 60
26    MULI   0, 0
27    LITI   0, 60
28    MULI   0, 0
29    SUBI   0, 0
30    STX    0, 0
31    POP    0, 1
32    LDA    1, 24
```

(생략)

```
53    LOD    1, 20
54    LOD    1, 24
55    LOD    1, 28
56    POP    0, 8
57    ADDR   0, printf
58    CAL    0, 0
59    LDA    1, -4
60    LITI   0, 0
61    STO    0, 0
62    RET    0, 0
63 .literal 12 "%dsec is %02d:%02d:%02d.\n"
```

- 어셈블리 프로그램을 시뮬레이터를 이용하여 실행하면 다음과 같은 결과가 나온다.

```

minjeong@minjeong-VirtualBox:~/mjeong/hw_final/simulator$ ./interp.exe a.asm
===== symbol =====
 1: printf  -1
 2: malloc  -2
 3: scanf   -3
 4: main     3
===== code =====
 0: INT      0,44
 1: SUP      0,3
 2: RET      0,0
 3: INT      0,32
 4: LDA      1,12
 5: LITI     0,54321
 6: STX      0,0
 7: POP      0,1
 8: LDA      1,16
 9: LOD      1,12
10: STX      0,0
11: POP      0,1
12: LDA      1,20
13: LOD      1,12
14: LITI     0,60
15: LITI     0,60
16: MULI     0,0
17: DIVI     0,0
18: STX      0,0
19: POP      0,1

```

(생략)

```

58: LITI     0,0
59: STO      0,0
60: RET      0,0

```

start execution

54321sec is 15:5:21.

end execution

```

minjeong@minjeong-VirtualBox:~/mjeong/hw_final/simulator$

```

test3.c를 gcc로 실행하면 다음과 같은 결과가 나온다. 두 개가 동일한 결과라는 것을 확인할 수 있다.

```

minjeong@minjeong-VirtualBox:~/mjeong/hw_final$ gcc -o gcc.exe test3.c
test3.c: In function 'main':
test3.c:16:2: warning: implicit declaration of function 'printf' [-Wimplicit-function-declaration]
 printf("%dsec is %02d:%02d:%02d.\n", s, hour, minute, second);
 ^
test3.c:16:2: warning: incompatible implicit declaration of built-in function 'printf'
test3.c:16:2: note: include '<stdio.h>' or provide a declaration of 'printf'
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$ ./gcc.exe
54321sec is 15:05:21.
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$

```

#### 4) test4.c

윤년인지 평년인지 판단하는 프로그램이다.

```
test4.c (~/mjeong/hw_final) - VIM
1 int yearCheck(int year){
2     int check;
3     if (year % 4 != 0){
4         check = 0;
5     }
6     else{
7         if (year % 100 != 0){
8             check = 1;
9         }
10        else {
11            if (year % 400 != 0){
12                check = 0;
13            }
14            else{
15                check = 1;
16            }
17        }
18    }
19    return check;
20 }
21
22 void output(int year, int check){
23     if (check == 0){
24         printf("%d is common year.\n", year);
25     }
26     else{
27         printf("%d is leap year.\n", year);
28     }
29 }
30
31 return;
32 }
33
34 int main(){
35     int year, check;
36     year = 1749;
37     check = yearCheck(year);
38     output(year, check);
39
40     return 0;
41 }
42 }
43
```

● 컴파일러를 통하여 만들어진 어셈블리 프로그램은 다음과 같다.

```
a.asm (~/mjeong/hw_final) - VIM
1  INT 0, 60
2  SUP 0, main
3  RET 0, 0
4 yearCheck:
5  INT 0, 20
6  LOD 1, 12
7  LITI 0, 4
8  MOD 0, 0
9  LITI 0, 0
10 NEQI 0, 0
11 JPC 0, L1
12 LDA 1, 16
13 LITI 0, 0
14 STX 0, 0
15 POP 0, 1
16 JMP 0, L2
17 L1:
18 LOD 1, 12
19 LITI 0, 100
20 MOD 0, 0
21 LITI 0, 0
22 NEQI 0, 0
23 JPC 0, L3
24 LDA 1, 16
25 LITI 0, 1
26 STX 0, 0
27 POP 0, 1
28 JMP 0, L4
29 L3:
30 LOD 1, 12
31 LITI 0, 400
32 MOD 0, 0
33 LITI 0, 0
34 NEQI 0, 0
35 JPC 0, L5
36 LDA 1, 16
37 LITI 0, 0
38 STX 0, 0
39 POP 0, 1
40 JMP 0, L6
41 L5:
42 LDA 1, 16
43 LITI 0, 1
44 STX 0, 0
45 POP 0, 1
46 L6:
47 L4:
48 L2:
49 LDA 1, -4
50 LOD 1, 16
51 STO 0, 0
52 RET 0, 0
53 output:
54 INT 0, 20
55 LOD 1, 16
56 LITI 0, 0
57 EQLI 0, 0
58 JPC 0, L7
59 INT 0, 12
60 LDA 0, 12
61 LOD 1, 12
62 POP 0, 5
63 ADDR 0, printf
64 CAL 0, 0
65 JMP 0, L8
66 L7:
67 INT 0, 12
68 LDA 0, 36
69 LOD 1, 12
70 POP 0, 5
71 ADDR 0, printf
72 CAL 0, 0
73 L8:
74 RET 0, 0
75 main:
76 INT 0, 20
77 LDA 1, 12
78 LITI 0, 1749
79 STX 0, 0
80 POP 0, 1
81 LDA 1, 16
82 INT 0, 16
83 LOD 1, 12
84 POP 0, 4
85 ADDR 0, yearCheck
86 CAL 0, 0
87 STX 0, 0
88 POP 0, 1
89 INT 0, 12
90 LOD 1, 12
91 LOD 1, 16
92 POP 0, 5
93 ADDR 0, output
94 CAL 0, 0
95 LDA 1, -4
96 LITI 0, 0
97 STO 0, 0
98 RET 0, 0
99 .literal 12 "%d is common year.\n"
100 .literal 36 "%d is leap year.\n"
```

- 어셈블리 프로그램을 시뮬레이터를 이용하여 실행하면 다음과 같은 결과가 나온다.

```

===== symbol =====
1: printf -1
2: malloc -2
3: scanf -3
4: main 64
5: yearCheck 3
6: L1 15
7: L2 41
8: L3 26
9: L4 41
10: L5 37
11: L6 41
12: output 45
13: L7 57
14: L8 63
===== code =====
0: INT 0,60
1: SUP 0,64
2: RET 0,0
3: INT 0,20
4: LOD 1,12
5: LITI 0,4
6: MOD 0,0
7: LITI 0,0
8: NEQI 0,0
9: JPC 0,15
10: LDA 1,16
11: LITI 0,0
12: STX 0,0
13: POP 0,1
14: JMP 0,41
15: LOD 1,12
16: LITI 0,100
17: MOD 0,0
18: LITI 0,0
19: NEQI 0,0
20: JPC 0,26
21: LDA 1,16
22: LITI 0,1
23: STX 0,0
24: POP 0,1
25: JMP 0,41
26: LOD 1,12
27: LITI 0,400
28: MOD 0,0
29: LITI 0,0
30: NEQI 0,0
31: JPC 0,37
32: LDA 1,16
33: LITI 0,0
34: STX 0,0
35: POP 0,1
36: JMP 0,41
37: LDA 1,16
38: LITI 0,1
39: STX 0,0
40: POP 0,1
41: LDA 1,-4
42: LOD 1,16
43: STO 0,0
44: RET 0,0
45: INT 0,20
46: LOD 1,16
47: LITI 0,0
48: EQLI 0,0
49: JPC 0,57
50: INT 0,12
51: LDA 0,12
52: LOD 1,12
53: POP 0,5
54: ADDR 0,-1
55: CAL 0,0
56: JMP 0,63
57: INT 0,12
58: LDA 0,36
59: LOD 1,12
60: POP 0,5
61: ADDR 0,-1
62: CAL 0,0
63: RET 0,0
64: INT 0,20
65: LDA 1,12
66: LITI 0,1749
67: STX 0,0
68: POP 0,1
69: LDA 1,16
70: INT 0,16
71: LOD 1,12
72: POP 0,4
73: ADDR 0,3
74: CAL 0,0
75: STX 0,0
76: POP 0,1
77: INT 0,12
78: LOD 1,12
79: LOD 1,16
80: POP 0,5
81: ADDR 0,45
82: CAL 0,0
83: LDA 1,-4
84: LITI 0,0
85: STO 0,0
86: RET 0,0
start execution
1749 is common year.
end execution
minjeong@minjeong-VirtualBox:~

```

test4.c를 gcc로 실행하면 다음과 같은 결과가 나온다. 두 개가 동일한 결과라는 것을 확인할 수 있다.

```

minjeong@minjeong-VirtualBox:~/njeong/hw_final$ gcc -o gcc.exe test4.c
test4.c: In function 'output':
test4.c:25:3: warning: implicit declaration of function 'printf' [-Wimplicit-function-declaration]
  printf("%d is common year.\n", year);
  ^
test4.c:25:3: warning: incompatible implicit declaration of built-in function 'printf'
test4.c:25:3: note: include '<stdio.h>' or provide a declaration of 'printf'
test4.c:28:3: warning: incompatible implicit declaration of built-in function 'printf'
  printf("%d is leap year.\n", year);
  ^
test4.c:28:3: note: include '<stdio.h>' or provide a declaration of 'printf'
minjeong@minjeong-VirtualBox:~/njeong/hw_final$ ./gcc.exe
1749 is common year.
minjeong@minjeong-VirtualBox:~/njeong/hw_final$

```

## 5) test5.c

```

test5.c (~/.mjeong/hw_final) - VIM
1 int main(){
2     int i=1, num;
3     num = 6;
4
5     while (i <= num){
6         printf("*");
7         if (i % 5 == 0){
8             printf("\n");
9         }
10        else{
11            ;
12        }
13        i++;
14    }
15    printf("\n");
16    return 0;
17 }

```

- 컴파일러를 통하여 만들어진 어셈블러 프로그램은 다음과 같다.

```

a.asm (~/.mjeong/hw_final) - VIM
1      INT    0, 32
2      SUP    0, main
3      RET    0, 0
4 main:
5      INT    0, 20
6      LDA    1, 16
7      LITI   0, 6
8      STX    0, 0
9      POP    0, 1
10 L2:
11      LOD    1, 12
12      LOD    1, 16
13      LEQI   0, 0
14      JPC    0, L3
15      INT    0, 12
16      LDA    0, 12
17      POP    0, 4
18      ADDR   0, printf
19      CAL    0, 0
20      LOD    1, 12
21      LITI   0, 5
22      MOD    0, 0
23      LITI   0, 0
24      EQLI   0, 0
25      JPC    0, L4
26      INT    0, 12
27      LDA    0, 16
28      POP    0, 4
29      ADDR   0, printf
30      CAL    0, 0
31      JMP    0, L5
32 L4:
33 L5:
34      LOD    1, 12
35      LDA    1, 12
36      LDX    0, 0
37      INCI   0, 0
38      STO    0, 0
39      POP    0, 1
40 L1:
41      JMP    0, L2
42 L3:
43      INT    0, 12
44      LDA    0, 24
45      POP    0, 4
46      ADDR   0, printf
47      CAL    0, 0
48      LDA    1, -4
49      LITI   0, 0
50      STO    0, 0
51      RET    0, 0
52 .literal   12  "*"
53 .literal   16  "\n"
54 .literal   24  "\n"
~
~
~

```



- 어셈블리 프로그램을 시뮬레이터를 이용하여 실행하면 다음과 같은 결과가 나온다.

```

===== symbol =====
 1: printf  -1
 2: malloc  -2
 3: scanf   -3
 4: main     3
 5: L2       8
 6: L3      36
 7: L4      29
 8: L5      29
 9: L1      35
===== code =====
 0: INT      0,32
 1: SUP      0,3
 2: RET      0,0
 3: INT      0,20
 4: LDA      1,16
 5: LITI     0,6
 6: STX      0,0
 7: POP      0,1
 8: LOD      1,12
 9: LOD      1,16
10: LEQI     0,0
11: JPC      0,36
12: INT      0,12
13: LDA      0,12
14: POP      0,4
15: ADDR     0,-1
16: CAL      0,0
17: LOD      1,12
18: LITI     0,5
19: MOD      0,0
20: LITI     0,0
21: EQLI     0,0
22: JPC      0,29
23: INT      0,12
24: LDA      0,16
25: POP      0,4
26: ADDR     0,-1
27: CAL      0,0
28: JMP      0,29
29: LOD      1,12
30: LDA      1,12
31: LDX      0,0
32: INCI     0,0
33: STO      0,0
34: POP      0,1
35: JMP      0,8
36: INT      0,12
37: LDA      0,24
38: POP      0,4
39: ADDR     0,-1
40: CAL      0,0
41: LDA      1,-4
42: LITI     0,0
43: STO      0,0
44: RET      0,0
start execution
*
*****
*
end execution
minjeong@minjeong-VirtualBox:

```

test5.c를 gcc로 실행하면 다음과 같은 결과가 나온다. 예상과 달리 시뮬레이터를 이용하여 실행한 결과가 이상하게 출력되었다. 그 원인에 대하여 고찰이 필요할 것 같다.

```

minjeong@minjeong-VirtualBox:~/njeong/hw_final$ gcc -o gcc.exe test5.c
test5.c: In function 'main':
test5.c:6:3: warning: implicit declaration of function 'printf' [-Wimplicit-function-declaration]
    printf("*");
    ^
test5.c:6:3: warning: incompatible implicit declaration of built-in function 'printf'
test5.c:6:3: note: include '<stdio.h>' or provide a declaration of 'printf'
test5.c:15:2: warning: incompatible implicit declaration of built-in function 'printf'
    printf("\n");
    ^
test5.c:15:2: note: include '<stdio.h>' or provide a declaration of 'printf'
minjeong@minjeong-VirtualBox:~/njeong/hw_final$ ./gcc.exe
*****
*
minjeong@minjeong-VirtualBox:~/njeong/hw_final$

```

# 6) test6.c

1~100까지의 숫자를 출력할 때 3의 배수는 '\*'로, 5의 배수는 '#'으로 출력하고 3과 5의 공배수는 정상 숫자로 출력하는 프로그램이다.

```
test6.c (~/mjeong/hw_final) - VIM
1 int main(){
2     int i, j;
3
4     for (i = 1; i < 100; i = i+10){
5         for (j = i; j <= i+9; j++){
6             if (j % 3 == 0 && j % 5 == 0){
7                 printf("%d\t", j);
8             }
9             else if (j % 3 == 0){
10                printf("*\t");
11            }
12            else if (j % 5 == 0){
13                printf("#\t");
14            }
15            else{
16                printf("%d\t", j);
17            }
18        }
19        printf("\n");
20    }
21    return 0;
22 }
```

● 컴파일러를 통하여 만들어진 어셈블러 프로그램은 다음과 같다.

```
a.asm (~/mjeong/hw_final) - VIM
1      INT    0, 52
2      SUP    0, main
3      RET     0, 0
4 main:
5      INT    0, 20
6      LDA    1, 12
7      LITI   0, 1
8      STX    0, 0
9      POP     0, 1
10 L2:      LOD    1, 12
11          LITI   0, 100
12          LSSI   0, 0
13          JPC    0, L3
14          LDA    1, 16
15          LOD    1, 12
16          STX    0, 0
17          POP     0, 1
18
19 L5:      LOD    1, 16
20          LOD    1, 12
21          LITI   0, 9
22          ADDI   0, 0
23          LEQI   0, 0
24          JPC    0, L6
25          LOD    1, 16
26          LITI   0, 3
27          MOD    0, 0
28          LITI   0, 0
29          EQLI   0, 0
30          JPCR   0, L7
31          LOD    1, 16
32          LITI   0, 5
33          MOD    0, 0
34          LITI   0, 0
35          EQLI   0, 0
36
37 L7:      JPC    0, L8
38          INT    0, 12
39          LDA    0, 12
40          LOD    1, 16
41          POP     0, 5
42          ADDR   0, printf
43          CAL    0, 0
44          JMP     0, L9
45
46 L8:      LOD    1, 16
47          LITI   0, 3
48          MOD    0, 0
49
50          LITI   0, 0
51          EQLI   0, 0
52          JPC    0, L10
53          INT    0, 12
54          LDA    0, 20
55          POP     0, 4
56          ADDR   0, printf
57          CAL    0, 0
58          JMP     0, L11
59 L10:     LOD    1, 16
60          LITI   0, 5
61          MOD    0, 0
62          LITI   0, 0
63          EQLI   0, 0
64          JPC    0, L12
65          INT    0, 12
66          LDA    0, 28
67          POP     0, 4
68          ADDR   0, printf
69          CAL    0, 0
70          JMP     0, L13
71
72 L12:     INT    0, 12
73          LDA    0, 36
74          LOD    1, 16
75          POP     0, 5
76          ADDR   0, printf
77          CAL    0, 0
78
79 L13:     LOD    1, 16
80          LDA    1, 16
81          LDX    0, 0
82          INCI   0, 0
83          STO    0, 0
84          POP     0, 1
85          JMP     0, L5
86
87 L4:      LOD    1, 16
88          LDA    1, 16
89          LDX    0, 0
90          INCI   0, 0
91          STO    0, 0
92          POP     0, 4
93          ADDR   0, printf
94          CAL    0, 0
95
96 L1:      LDA    1, 12
97          LOD    1, 12
98          LSSI   0, 0
99          LITI   0, 10
100         ADDI   0, 0
101         STX    0, 0
102         POP     0, 1
103         JMP     0, L2
104 L3:      LDA    1, -4
105         LITI   0, 0
106         STO    0, 0
107         RET     0, 0
108
109 .literal 12  "%d\t"
110 .literal 20  "*\t"
111 .literal 28  "#\t"
112 .literal 36  "%d\t"
113 .literal 44  "\n"
```

- 어셈블리 프로그램을 시뮬레이터를 이용하여 실행하면 다음과 같은 결과가 나온다.

```

===== symbol =====
1: printf -1
2: malloc -2
3: scanf -3
4: main 3
5: L2 8
6: L3 90
7: L5 16
8: L6 78
9: L7 33
10: L8 41
11: L9 71
12: L10 53
13: L11 71
14: L12 65
15: L13 71
16: L4 71
17: L1 83
===== code =====
0: INT 0,52
1: SUP 0,3
2: RET 0,0
3: INT 0,20
4: LDA 1,12
5: LITI 0,1
6: STX 0,0
7: POP 0,1
8: LOD 1,12
9: LITI 0,100
10: LSSI 0,0
11: JPC 0,90
12: LDA 1,16
13: LOD 1,12
14: STX 0,0
15: POP 0,1
16: LOD 1,16
17: LOD 1,12
18: LITI 0,9
19: ADDI 0,0
20: LEQI 0,0
21: JPC 0,78
22: LOD 1,16
23: LITI 0,3
24: MOD 0,0
25: LITI 0,0
26: EQLI 0,0

68: POP 0,5
69: ADDR 0,-1
70: CAL 0,0
71: LOD 1,16
72: LDA 1,16
73: LDX 0,0
74: INCI 0,0
75: STO 0,0
76: POP 0,1
77: JMP 0,16
78: INT 0,12
79: LDA 0,44
80: POP 0,4
81: ADDR 0,-1
82: CAL 0,0
83: LDA 1,12
84: LOD 1,12
85: LITI 0,10
86: ADDI 0,0
87: STX 0,0
88: POP 0,1
89: JMP 0,8
90: LDA 1,-4
91: LITI 0,0
92: STO 0,0
93: RET 0,0

start execution
1 2 * 4 # * 7 8 * #
11 * 13 14 15 16 17 * 19 #
* 22 23 * # 26 * 28 29 30
31 32 * 34 # * 37 38 * #
41 * 43 44 45 46 47 * 49 #
* 52 53 * # 56 * 58 59 60
61 62 * 64 # * 67 68 * #
71 * 73 74 75 76 77 * 79 #
* 82 83 * # 86 * 88 89 90
91 92 * 94 # * 97 98 * #
end execution

```

test6.c를 gcc로 실행하면 다음과 같은 결과가 나온다. 두 개가 동일한 결과라는 것을 확인할 수 있다.

```

minjeong@minjeong-VirtualBox:~/mjeong/hw_final$ gcc -o gcc.exe test6.c
test6.c: In function 'main':
test6.c:7:5: warning: implicit declaration of function 'printf' [-Wimplicit-function-declaration]
printf("%d\t", j);
^
test6.c:7:5: warning: incompatible implicit declaration of built-in function 'printf'
test6.c:7:5: note: include '<stdio.h>' or provide a declaration of 'printf'
test6.c:10:5: warning: incompatible implicit declaration of built-in function 'printf'
printf("%t\t");
^
test6.c:10:5: note: include '<stdio.h>' or provide a declaration of 'printf'
test6.c:13:5: warning: incompatible implicit declaration of built-in function 'printf'
printf("#t\t");
^
test6.c:13:5: note: include '<stdio.h>' or provide a declaration of 'printf'
test6.c:16:5: warning: incompatible implicit declaration of built-in function 'printf'
printf("%d\t", j);
^
test6.c:16:5: note: include '<stdio.h>' or provide a declaration of 'printf'
test6.c:19:3: warning: incompatible implicit declaration of built-in function 'printf'
printf("\n");
^
test6.c:19:3: note: include '<stdio.h>' or provide a declaration of 'printf'
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$ ./gcc.exe
1 2 * 4 # * 7 8 * #
11 * 13 14 15 16 17 * 19 #
* 22 23 * # 26 * 28 29 30
31 32 * 34 # * 37 38 * #
41 * 43 44 45 46 47 * 49 #
* 52 53 * # 56 * 58 59 60
61 62 * 64 # * 67 68 * #
71 * 73 74 75 76 77 * 79 #
* 82 83 * # 86 * 88 89 90
91 92 * 94 # * 97 98 * #
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$

```

7) test7.c

0부터 49까지의 숫자를 10개단위로 나눠서 출력하는 프로그램이다.

```
test7.c (~/mjeong/hw_final) - VIM
1 int main(){
2     int i, j;
3     for (i = 0; i < 10; i++){
4         for (j = i; j < 50; j = j+10){
5             printf("%d ", j);
6         }
7         printf("\n");
8     }
9
10    return 0;
11 }
```

● 컴파일러를 통하여 만들어진 어셈블러 프로그램은 다음과 같다.

```
a.asm (~/mjeong/hw_final) - VIM
1      INT    0, 28
2      SUP    0, main
3      RET    0, 0
4  main:
5      INT    0, 20
6      LDA    1, 12
7      LITI   0, 0
8      STX    0, 0
9      POP    0, 1
10 L2:
11      LOD    1, 12
12      LITI   0, 10
13      LSSI   0, 0
14      JPC    0, L3
15      LDA    1, 16
16      LOD    1, 12
17      STX    0, 0
18      POP    0, 1
19 L5:
20      LOD    1, 16
21      LITI   0, 50
22      LSSI   0, 0
23      JPC    0, L6
24      INT    0, 12
25      LDA    0, 12
26      LOD    1, 16
27      POP    0, 5
28      ADDR   0, printf
29      CAL    0, 0
30 L4:
31      LDA    1, 16
32      LOD    1, 16
33      LITI   0, 10
34      ADDI   0, 0
35      STX    0, 0
36      POP    0, 1
37      JMP    0, L5
38 L6:
39      INT    0, 12
40      LDA    0, 20
41      POP    0, 4
42      ADDR   0, printf
43      CAL    0, 0
44 L1:
45      LOD    1, 12
46      LDA    1, 12
47      LDX    0, 0
48      INCI   0, 0
49      STO    0, 0
50      POP    0, 1
51      JMP    0, L2
52 L3:
53      LDA    1, -4
54      LITI   0, 0
55      STO    0, 0
56      RET    0, 0
57 .literal   12  "%d "
58 .literal   20  "\n"
```

● 어셈블러 프로그램을 시뮬레이터를 이용하여 실행하면 다음과 같은 결과가 나온다.



```

===== symbol =====
1: printf -1
2: malloc -2
3: scanf -3
4: main 3
5: L2 8
6: L3 45
7: L5 16
8: L6 33
9: L4 26
10: L1 38
===== code =====
0: INT 0,28
1: SUP 0,3
2: RET 0,0
3: INT 0,20
4: LDA 1,12
5: LITI 0,0
6: STX 0,0
7: POP 0,1
8: LOD 1,12
9: LITI 0,10
10: LSSI 0,0
11: JPC 0,45
12: LDA 1,16
13: LOD 1,12
14: STX 0,0
15: POP 0,1
16: LOD 1,16
17: LITI 0,50
18: LSSI 0,0
19: JPC 0,33
20: INT 0,12
21: LDA 0,12
22: LOD 1,16
23: POP 0,5
24: ADDR 0,-1
25: CAL 0,0
26: LDA 1,16
27: LOD 1,16
28: LITI 0,10
29: ADDI 0,0
30: STX 0,0
31: POP 0,1
32: JMP 0,16
33: INT 0,12
34: LDA 0,20
35: POP 0,4
36: ADDR 0,-1
37: CAL 0,0
38: LOD 1,12
39: LDA 1,12
40: LDX 0,0
41: INCI 0,0
42: STO 0,0
43: POP 0,1
44: JMP 0,8
45: LDA 1,-4
46: LITI 0,0
47: STO 0,0
48: RET 0,0
start execution
0 10 20 30 40
1 11 21 31 41
2 12 22 32 42
3 13 23 33 43
4 14 24 34 44
5 15 25 35 45
6 16 26 36 46
7 17 27 37 47
8 18 28 38 48
9 19 29 39 49
end execution

```

test7.c를 gcc로 실행하면 다음과 같은 결과가 나온다. 두 개가 동일한 결과라는 것을 확인할 수 있다.

```

minjeong@minjeong-VirtualBox:~/mjeong/hw_final$ gcc -o gcc.exe test7.c
test7.c: In function 'main':
test7.c:5:4: warning: implicit declaration of function 'printf' [-Wimplicit-function-declaration]
    printf("%d ", j);
    ^
test7.c:5:4: warning: incompatible implicit declaration of built-in function 'printf'
test7.c:5:4: note: include '<stdio.h>' or provide a declaration of 'printf'
test7.c:7:3: warning: incompatible implicit declaration of built-in function 'printf'
    printf("\n");
    ^
test7.c:7:3: note: include '<stdio.h>' or provide a declaration of 'printf'
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$ ./gcc.exe
0 10 20 30 40
1 11 21 31 41
2 12 22 32 42
3 13 23 33 43
4 14 24 34 44
5 15 25 35 45
6 16 26 36 46
7 17 27 37 47
8 18 28 38 48
9 19 29 39 49

```



## 8) test8.c

삼중 for문을 이용하여 구구단을 출력한 프로그램이다.

```
test8.c (~/mjeong/hw_final) - VIM
1 int main(){
2     int i, j, k;
3     for (i = 2; i < 9; i = i+4){
4         for (j = 1; j <= 9; j++){
5             for (k = i; k-i < 4; k++){
6                 printf("%d * %d = %d\t", k, j, k*j);
7             }
8             printf("\n");
9         }
10        printf("\n");
11    }
12
13 }
```

● 컴파일러를 통하여 만들어진 어셈블러 프로그램은 다음과 같다.

```
a.asm (~/mjeong/hw_final) - VI
1      INT    0, 48
2      SUP    0, main
3      RET    0, 0
4 main:
5      INT    0, 24
6      LDA    1, 12
7      LITI   0, 2
8      STX    0, 0
9      POP    0, 1
10 L2:
11      LOD    1, 12
12      LITI   0, 9
13      LSSI   0, 0
14      JPC    0, L3
15      LDA    1, 16
16      LITI   0, 1
17      STX    0, 0
18      POP    0, 1
19 L5:
20      LOD    1, 16
21      LITI   0, 9
22      LEQI   0, 0
23      JPC    0, L6
24      LDA    1, 20
25      LOD    1, 12
26      STX    0, 0
27      POP    0, 1
28 L8:
29      LOD    1, 20
30      LOD    1, 12
31      SUBI   0, 0
32      SUBI   0, 4
33      LSSI   0, 0
34      JPC    0, L9
35      INT    0, 12
36      LDA    0, 12
37      LOD    1, 20
38      LOD    1, 16
39      LOD    1, 20
40      LOD    1, 16
41      MULI   0, 0
42      POP    0, 7
43      ADDR   0, printf
44      CAL    0, 0
45 L7:
46      LOD    1, 20
47      LDA    1, 20
48      LDX    0, 0
49      INCI   0, 0
50      STO    0, 0
51      POP    0, 1
52      JMP    0, L8
53 L9:
54      INT    0, 12
55      LDA    0, 32
56      POP    0, 4
57      ADDR   0, printf
58      CAL    0, 0
59 L4:
60      LOD    1, 16
61      LDA    1, 16
62      LDX    0, 0
63      INCI   0, 0
64      STO    0, 0
65      POP    0, 1
66      JMP    0, L5
67 L6:
68      INT    0, 12
69      LDA    0, 40
70      POP    0, 4
71      ADDR   0, printf
72      CAL    0, 0
73 L1:
74      LDA    1, 12
75      LOD    1, 12
76      LITI   0, 4
77      ADDI   0, 0
78      STX    0, 0
79      POP    0, 1
80      JMP    0, L2
81 L3:
82      RET    0, 0
83 .literal   12  "%d * %d = %d\t"
84 .literal   32  "\n"
85 .literal   40  "\n"
```

● 어셈블러 프로그램을 시뮬레이터를 이용하여 실행하면 다음과 같은 결과가 나온다.

```

===== symbol =====
1: printf -1
2: malloc -2
3: scanf -3
4: main 3
5: L2 8
6: L3 71
7: L5 16
8: L6 59
9: L8 24
10: L9 47
11: L7 40
12: L4 52
13: L1 64
===== code =====
0: INT 0,48
1: SUP 0,3
2: RET 0,0
3: INT 0,24
4: LDA 1,12
5: LITI 0,2
6: STX 0,0
7: POP 0,1
8: LOD 1,12
9: LITI 0,9
10: LSSI 0,0
11: JPC 0,71
12: LDA 1,16
13: LITI 0,1
14: STX 0,0
15: POP 0,1
16: LOD 1,16
17: LITI 0,9
18: LEQI 0,0
19: JPC 0,59
20: LDA 1,20
21: LOD 1,12
22: STX 0,0
23: POP 0,1
24: LOD 1,20
25: LOD 1,12
26: SUBI 0,0
27: LITI 0,4
28: LSSI 0,0
29: JPC 0,47
30: INT 0,12
31: LDA 0,12
32: LOD 1,20
33: LOD 1,16
34: LOD 1,20
35: LOD 1,16
36: MULI 0,0
37: POP 0,7
38: ADDR 0,-1
39: CAL 0,0
40: LOD 1,20
41: LDA 1,20
42: LDX 0,0
43: INCI 0,0
44: STO 0,0
45: POP 0,1
46: JMP 0,24
47: INT 0,12
48: LDA 0,32
49: POP 0,4
50: ADDR 0,-1
51: CAL 0,0
52: LOD 1,16
53: LDA 1,16
54: LDX 0,0
55: INCI 0,0
56: STO 0,0
57: POP 0,1
58: JMP 0,16
59: INT 0,12
60: LDA 0,40
61: POP 0,4
62: ADDR 0,-1
63: CAL 0,0
64: LDA 1,12
65: LOD 1,12
66: LITI 0,4
67: ADDI 0,0
68: STX 0,0
69: POP 0,1
70: JMP 0,8
71: RET 0,0
start execution
2 * 1 = 2    3 * 1 = 3    4 * 1 = 4    5 * 1 = 5
2 * 2 = 4    3 * 2 = 6    4 * 2 = 8    5 * 2 = 10
2 * 3 = 6    3 * 3 = 9    4 * 3 = 12    5 * 3 = 15
2 * 4 = 8    3 * 4 = 12    4 * 4 = 16    5 * 4 = 20
2 * 5 = 10   3 * 5 = 15    4 * 5 = 20    5 * 5 = 25
2 * 6 = 12   3 * 6 = 18    4 * 6 = 24    5 * 6 = 30
2 * 7 = 14   3 * 7 = 21    4 * 7 = 28    5 * 7 = 35
2 * 8 = 16   3 * 8 = 24    4 * 8 = 32    5 * 8 = 40
2 * 9 = 18   3 * 9 = 27    4 * 9 = 36    5 * 9 = 45
6 * 1 = 6    7 * 1 = 7    8 * 1 = 8    9 * 1 = 9
6 * 2 = 12   7 * 2 = 14   8 * 2 = 16   9 * 2 = 18
6 * 3 = 18   7 * 3 = 21   8 * 3 = 24   9 * 3 = 27
6 * 4 = 24   7 * 4 = 28   8 * 4 = 32   9 * 4 = 36
6 * 5 = 30   7 * 5 = 35   8 * 5 = 40   9 * 5 = 45
6 * 6 = 36   7 * 6 = 42   8 * 6 = 48   9 * 6 = 54
6 * 7 = 42   7 * 7 = 49   8 * 7 = 56   9 * 7 = 63
6 * 8 = 48   7 * 8 = 56   8 * 8 = 64   9 * 8 = 72
6 * 9 = 54   7 * 9 = 63   8 * 9 = 72   9 * 9 = 81
end execution
minjeong@minjeong-VirtualBox:~/mieong/hw_final/simulators$

```

test8.c를 gcc로 실행하면 다음과 같은 결과가 나온다. 두 개가 동일한 결과라는 것을 확인할 수 있다.

```

minjeong@minjeong-VirtualBox:~/mieong/hw_final$ gcc -o gcc.exe test8.c
test8.c: In function 'main':
test8.c:6:5: warning: implicit declaration of function 'printf' [-Wimplicit-function-declaration]
    printf("%d * %d = %d\t", k, j, k*j);
    ^
test8.c:6:5: warning: incompatible implicit declaration of built-in function 'printf'
test8.c:6:5: note: include '<stdio.h>' or provide a declaration of 'printf'
test8.c:8:4: warning: incompatible implicit declaration of built-in function 'printf'
    printf("\n");
    ^
test8.c:8:4: note: include '<stdio.h>' or provide a declaration of 'printf'
test8.c:10:3: warning: incompatible implicit declaration of built-in function 'printf'
    printf("\n");
    ^
test8.c:10:3: note: include '<stdio.h>' or provide a declaration of 'printf'
minjeong@minjeong-VirtualBox:~/mieong/hw_final$ ./gcc.exe
2 * 1 = 2    3 * 1 = 3    4 * 1 = 4    5 * 1 = 5
2 * 2 = 4    3 * 2 = 6    4 * 2 = 8    5 * 2 = 10
2 * 3 = 6    3 * 3 = 9    4 * 3 = 12    5 * 3 = 15
2 * 4 = 8    3 * 4 = 12    4 * 4 = 16    5 * 4 = 20
2 * 5 = 10   3 * 5 = 15    4 * 5 = 20    5 * 5 = 25
2 * 6 = 12   3 * 6 = 18    4 * 6 = 24    5 * 6 = 30
2 * 7 = 14   3 * 7 = 21    4 * 7 = 28    5 * 7 = 35
2 * 8 = 16   3 * 8 = 24    4 * 8 = 32    5 * 8 = 40
2 * 9 = 18   3 * 9 = 27    4 * 9 = 36    5 * 9 = 45

6 * 1 = 6    7 * 1 = 7    8 * 1 = 8    9 * 1 = 9
6 * 2 = 12   7 * 2 = 14   8 * 2 = 16   9 * 2 = 18
6 * 3 = 18   7 * 3 = 21   8 * 3 = 24   9 * 3 = 27
6 * 4 = 24   7 * 4 = 28   8 * 4 = 32   9 * 4 = 36
6 * 5 = 30   7 * 5 = 35   8 * 5 = 40   9 * 5 = 45
6 * 6 = 36   7 * 6 = 42   8 * 6 = 48   9 * 6 = 54
6 * 7 = 42   7 * 7 = 49   8 * 7 = 56   9 * 7 = 63
6 * 8 = 48   7 * 8 = 56   8 * 8 = 64   9 * 8 = 72
6 * 9 = 54   7 * 9 = 63   8 * 9 = 72   9 * 9 = 81
minjeong@minjeong-VirtualBox:~/mieong/hw_final$

```

# 9) test9.c

배열 안에 있는 숫자의 개수를 세는 프로그램이다.

```
test9.c (~/mjeong/hw_final) - VIM
1 int main(){
2     int i, j, k;
3     int ary[10],count[10];
4     ary[0] = 2;
5     ary[1] = 8;
6     ary[2] = 5;
7     ary[3] = 8;
8     ary[4] = 2;
9     ary[5] = 3;
10    ary[6] = 3;
11    ary[7] = 9;
12    ary[8] = 1;
13    ary[9] = 3;
14    for (i = 0; i < 10; i++)
15    {
16        count[i] = 0;
17    }
18
19    for (i = 0; i < 10; i++)
20    {
21        for (j = 0; j <= sizeof(ary) / sizeof(int); j++)
22        {
23            if (i + 1 == ary[j]) { count[i]++; }
24        }
25    }
26    for (k = 0; k < 10; k++)
27    {
28        printf("The number of %d is %d\n", k + 1, count[k]);
29    }
30
31    return 0;
32 }
```

● 컴파일러를 통하여 만들어진 어셈블러 프로그램은 다음과 같다.

```
a.asm (~/mjeong/hw_final)
1  INT 0, 40 28
2  SUP 0, main 29
3  RET 0, 0 30
4  main: 31
5  INT 0, 104 32
6  LDA 1, 24 33
7  LITI 0, 0 34
8  LITI 0, 4 35
9  MULI 0, 0 36
10 OFFSET 0, 0 37
11 LITI 0, 2 38
12 STX 0, 0 39
13 POP 0, 1 40
14 LDA 1, 24 41
15 LITI 0, 1 42
16 LITI 0, 4 43
17 MULI 0, 0 44
18 OFFSET 0, 0 45
19 LITI 0, 8 46
20 STX 0, 0 47
21 POP 0, 1 48
22 LDA 1, 24 49
23 LITI 0, 2 50
24 LITI 0, 4 51
25 MULI 0, 0 52
26 OFFSET 0, 0 53
27 LITI 0, 5 54
28 STX 0, 0 55
29 POP 0, 1 56
30 LDA 1, 24 57
31 LITI 0, 3 58
32 LITI 0, 4 59
33 MULI 0, 0 60
34 OFFSET 0, 0 61
35 LITI 0, 8 62
36 STX 0, 0 63
37 POP 0, 1 64
38 LDA 1, 24 65
39 LITI 0, 4 66
40 LITI 0, 4 67
41 MULI 0, 0 68
42 OFFSET 0, 0 69
43 LITI 0, 2 70
44 STX 0, 0 71
45 POP 0, 1 72
46 LDA 1, 24 73
47 LITI 0, 5 74
48 LITI 0, 4 75
49 MULI 0, 0 76
50 OFFSET 0, 0 77
51 LITI 0, 3 78
52 STX 0, 0 79
53 POP 0, 1 80
54 LDA 1, 24 81
55 LITI 0, 6 82
56 LITI 0, 4 83
57 STX 0, 0 84
58 POP 0, 1 85
59 LDA 1, 24 86
60 LITI 0, 3 87
61 LITI 0, 4 88
62 MULI 0, 0 89
63 OFFSET 0, 0 90
64 LITI 0, 3 91
65 STX 0, 0 92
66 POP 0, 1 93
67 LDA 1, 24 94
68 LITI 0, 9 95
69 LITI 0, 4 96
70 MULI 0, 0 97
71 OFFSET 0, 0 98
72 LITI 0, 3 99
73 STX 0, 0 100
74 POP 0, 1 101
75 LDA 1, 24 102
76 LITI 0, 9 103
77 LITI 0, 4 104
78 MULI 0, 0 105
79 OFFSET 0, 0 106
80 LITI 0, 3 107
81 STX 0, 0 108
82 POP 0, 1 109
83 LDA 1, 24 110
84 LITI 0, 9 111
85 LITI 0, 4 112
86 MULI 0, 0 113
87 OFFSET 0, 0 114
88 LITI 0, 3 115
89 STX 0, 0 116
90 POP 0, 1 117
91 LDA 1, 24 118
92 LITI 0, 9 119
93 LITI 0, 4 120
94 MULI 0, 0 121
95 OFFSET 0, 0 122
96 LITI 0, 3 123
97 STX 0, 0 124
98 POP 0, 1 125
99 LDA 1, 24 126
100 LITI 0, 9 127
101 LITI 0, 4 128
102 MULI 0, 0 129
103 OFFSET 0, 0 130
104 LITI 0, 3 131
105 STX 0, 0 132
106 POP 0, 1 133
107 LDA 1, 24 134
108 LITI 0, 9 135
109 LITI 0, 4 136
110 MULI 0, 0 137
111 OFFSET 0, 0 138
112 LITI 0, 3 139
113 STX 0, 0 140
114 POP 0, 1 141
115 LDA 1, 24 142
116 LITI 0, 9 143
117 LITI 0, 4 144
118 MULI 0, 0 145
119 OFFSET 0, 0 146
120 LITI 0, 3 147
121 STX 0, 0 148
122 POP 0, 1 149
123 LDA 1, 24 150
124 LITI 0, 9 151
125 LITI 0, 4 152
126 MULI 0, 0 153
127 OFFSET 0, 0 154
128 LITI 0, 3 155
129 STX 0, 0 156
130 POP 0, 1 157
131 LDA 1, 24 158
132 LITI 0, 9 159
133 LITI 0, 4 160
134 MULI 0, 0 161
135 OFFSET 0, 0 162
136 LITI 0, 3 163
137 STX 0, 0 164
138 POP 0, 1 165
139 LDA 1, 24 166
140 LITI 0, 9 167
141 LITI 0, 4 168
142 MULI 0, 0 169
143 OFFSET 0, 0 170
144 LITI 0, 3 171
145 STX 0, 0 172
146 POP 0, 1 173
147 LDA 1, 24 174
148 LITI 0, 9 175
149 LITI 0, 4 176
150 MULI 0, 0 177
151 OFFSET 0, 0 178
152 LITI 0, 3 179
153 STX 0, 0 180
154 POP 0, 1 181
155 LDA 1, 24 182
156 LITI 0, 9 183
157 LITI 0, 4 184
158 MULI 0, 0 185
159 OFFSET 0, 0 186
160 LITI 0, 3 187
161 STX 0, 0 188
162 POP 0, 1 189
163 LDA 1, 24 190
164 LITI 0, 9 191
165 LITI 0, 4 192
166 MULI 0, 0 193
167 OFFSET 0, 0 194
168 LITI 0, 3 195
169 STX 0, 0 196
170 POP 0, 1 197
171 LDA 1, 24 198
172 LITI 0, 9 199
173 LITI 0, 4 200
174 MULI 0, 0 201
175 OFFSET 0, 0 202
176 LITI 0, 3 203
177 STX 0, 0 204
178 POP 0, 1 205
179 LDA 1, 24 206
180 LITI 0, 9 207
181 LITI 0, 4 208
182 MULI 0, 0 209
183 OFFSET 0, 0 210
184 LITI 0, 3 211
185 STX 0, 0 212
186 POP 0, 1 213
187 LDA 1, 24 214
188 LITI 0, 9 215
189 LITI 0, 4 216
190 MULI 0, 0 217
191 OFFSET 0, 0 218
192 LITI 0, 3 219
193 STX 0, 0 220
194 POP 0, 1 221
195 LDA 1, 24 222
196 LITI 0, 9 223
197 LITI 0, 4 224
198 MULI 0, 0 225
199 OFFSET 0, 0 226
200 LITI 0, 3 227
201 STX 0, 0 228
202 POP 0, 1 229
203 LDA 1, 24 230
204 LITI 0, 9 231
205 LITI 0, 4 232
206 MULI 0, 0 233
207 OFFSET 0, 0 234
208 LITI 0, 3 235
209 STX 0, 0 236
210 POP 0, 1 237
211 LDA 1, 24 238
212 LITI 0, 9 239
213 LITI 0, 4 240
214 MULI 0, 0 241
215 OFFSET 0, 0 242
216 LITI 0, 3 243
217 STX 0, 0 244
218 POP 0, 1 245
219 LDA 1, 24 246
220 LITI 0, 9 247
221 LITI 0, 4 248
222 MULI 0, 0 249
223 OFFSET 0, 0 250
224 LITI 0, 3 251
225 STX 0, 0 252
226 POP 0, 1 253
227 LDA 1, 24 254
228 LITI 0, 9 255
229 LITI 0, 4 256
230 MULI 0, 0 257
231 OFFSET 0, 0 258
232 LITI 0, 3 259
233 STX 0, 0 260
234 POP 0, 1 261
235 LDA 1, 24 262
236 LITI 0, 9 263
237 LITI 0, 4 264
238 MULI 0, 0 265
239 OFFSET 0, 0 266
240 LITI 0, 3 267
241 STX 0, 0 268
242 POP 0, 1 269
243 LDA 1, 24 270
244 LITI 0, 9 271
245 LITI 0, 4 272
246 MULI 0, 0 273
247 OFFSET 0, 0 274
248 LITI 0, 3 275
249 STX 0, 0 276
250 POP 0, 1 277
251 LDA 1, 24 278
252 LITI 0, 9 279
253 LITI 0, 4 280
254 MULI 0, 0 281
255 OFFSET 0, 0 282
256 LITI 0, 3 283
257 STX 0, 0 284
258 POP 0, 1 285
259 LDA 1, 24 286
260 LITI 0, 9 287
261 LITI 0, 4 288
262 MULI 0, 0 289
263 OFFSET 0, 0 290
264 LITI 0, 3 291
265 STX 0, 0 292
266 POP 0, 1 293
267 LDA 1, 24 294
268 LITI 0, 9 295
269 LITI 0, 4 296
270 MULI 0, 0 297
271 OFFSET 0, 0 298
272 LITI 0, 3 299
273 STX 0, 0 300
274 POP 0, 1 301
275 LDA 1, 24 302
276 LITI 0, 9 303
277 LITI 0, 4 304
278 MULI 0, 0 305
279 OFFSET 0, 0 306
280 LITI 0, 3 307
281 STX 0, 0 308
282 POP 0, 1 309
283 LDA 1, 24 310
284 LITI 0, 9 311
285 LITI 0, 4 312
286 MULI 0, 0 313
287 OFFSET 0, 0 314
288 LITI 0, 3 315
289 STX 0, 0 316
290 POP 0, 1 317
291 LDA 1, 24 318
292 LITI 0, 9 319
293 LITI 0, 4 320
294 MULI 0, 0 321
295 OFFSET 0, 0 322
296 LITI 0, 3 323
297 STX 0, 0 324
298 POP 0, 1 325
299 LDA 1, 24 326
300 LITI 0, 9 327
301 LITI 0, 4 328
302 MULI 0, 0 329
303 OFFSET 0, 0 330
304 LITI 0, 3 331
305 STX 0, 0 332
306 POP 0, 1 333
307 LDA 1, 24 334
308 LITI 0, 9 335
309 LITI 0, 4 336
310 MULI 0, 0 337
311 OFFSET 0, 0 338
312 LITI 0, 3 339
313 STX 0, 0 340
314 POP 0, 1 341
315 LDA 1, 24 342
316 LITI 0, 9 343
317 LITI 0, 4 344
318 MULI 0, 0 345
319 OFFSET 0, 0 346
320 LITI 0, 3 347
321 STX 0, 0 348
322 POP 0, 1 349
323 LDA 1, 24 350
324 LITI 0, 9 351
325 LITI 0, 4 352
326 MULI 0, 0 353
327 OFFSET 0, 0 354
328 LITI 0, 3 355
329 STX 0, 0 356
330 POP 0, 1 357
331 LDA 1, 24 358
332 LITI 0, 9 359
333 LITI 0, 4 360
334 MULI 0, 0 361
335 OFFSET 0, 0 362
336 LITI 0, 3 363
337 STX 0, 0 364
338 POP 0, 1 365
339 LDA 1, 24 366
340 LITI 0, 9 367
341 LITI 0, 4 368
342 MULI 0, 0 369
343 OFFSET 0, 0 370
344 LITI 0, 3 371
345 STX 0, 0 372
346 POP 0, 1 373
347 LDA 1, 24 374
348 LITI 0, 9 375
349 LITI 0, 4 376
350 MULI 0, 0 377
351 OFFSET 0, 0 378
352 LITI 0, 3 379
353 STX 0, 0 380
354 POP 0, 1 381
355 LDA 1, 24 382
356 LITI 0, 9 383
357 LITI 0, 4 384
358 MULI 0, 0 385
359 OFFSET 0, 0 386
360 LITI 0, 3 387
361 STX 0, 0 388
362 POP 0, 1 389
363 LDA 1, 24 390
364 LITI 0, 9 391
365 LITI 0, 4 392
366 MULI 0, 0 393
367 OFFSET 0, 0 394
368 LITI 0, 3 395
369 STX 0, 0 396
370 POP 0, 1 397
371 LDA 1, 24 398
372 LITI 0, 9 399
373 LITI 0, 4 400
374 MULI 0, 0 401
375 OFFSET 0, 0 402
376 LITI 0, 3 403
377 STX 0, 0 404
378 POP 0, 1 405
379 LDA 1, 24 406
380 LITI 0, 9 407
381 LITI 0, 4 408
382 MULI 0, 0 409
383 OFFSET 0, 0 410
384 LITI 0, 3 411
385 STX 0, 0 412
386 POP 0, 1 413
387 LDA 1, 24 414
388 LITI 0, 9 415
389 LITI 0, 4 416
390 MULI 0, 0 417
391 OFFSET 0, 0 418
392 LITI 0, 3 419
393 STX 0, 0 420
394 POP 0, 1 421
395 LDA 1, 24 422
396 LITI 0, 9 423
397 LITI 0, 4 424
398 MULI 0, 0 425
399 OFFSET 0, 0 426
400 LITI 0, 3 427
401 STX 0, 0 428
402 POP 0, 1 429
403 LDA 1, 24 430
404 LITI 0, 9 431
405 LITI 0, 4 432
406 MULI 0, 0 433
407 OFFSET 0, 0 434
408 LITI 0, 3 435
409 STX 0, 0 436
410 POP 0, 1 437
411 LDA 1, 24 438
412 LITI 0, 9 439
413 LITI 0, 4 440
414 MULI 0, 0 441
415 OFFSET 0, 0 442
416 LITI 0, 3 443
417 STX 0, 0 444
418 POP 0, 1 445
419 LDA 1, 24 446
420 LITI 0, 9 447
421 LITI 0, 4 448
422 MULI 0, 0 449
423 OFFSET 0, 0 450
424 LITI 0, 3 451
425 STX 0, 0 452
426 POP 0, 1 453
427 LDA 1, 24 454
428 LITI 0, 9 455
429 LITI 0, 4 456
430 MULI 0, 0 457
431 OFFSET 0, 0 458
432 LITI 0, 3 459
433 STX 0, 0 460
434 POP 0, 1 461
435 LDA 1, 24 462
436 LITI 0, 9 463
437 LITI 0, 4 464
438 MULI 0, 0 465
439 OFFSET 0, 0 466
440 LITI 0, 3 467
441 STX 0, 0 468
442 POP 0, 1 469
443 LDA 1, 24 470
444 LITI 0, 9 471
445 LITI 0, 4 472
446 MULI 0, 0 473
447 OFFSET 0, 0 474
448 LITI 0, 3 475
449 STX 0, 0 476
450 POP 0, 1 477
451 LDA 1, 24 478
452 LITI 0, 9 479
453 LITI 0, 4 480
454 MULI 0, 0 481
455 OFFSET 0, 0 482
456 LITI 0, 3 483
457 STX 0, 0 484
458 POP 0, 1 485
459 LDA 1, 24 486
460 LITI 0, 9 487
461 LITI 0, 4 488
462 MULI 0, 0 489
463 OFFSET 0, 0 490
464 LITI 0, 3 491
465 STX 0, 0 492
466 POP 0, 1 493
467 LDA 1, 24 494
468 LITI 0, 9 495
469 LITI 0, 4 496
470 MULI 0, 0 497
471 OFFSET 0, 0 498
472 LITI 0, 3 499
473 STX 0, 0 500
474 POP 0, 1 501
475 LDA 1, 24 502
476 LITI 0, 9 503
477 LITI 0, 4 504
478 MULI 0, 0 505
479 OFFSET 0, 0 506
480 LITI 0, 3 507
481 STX 0, 0 508
482 POP 0, 1 509
483 LDA 1, 24 510
484 LITI 0, 9 511
485 LITI 0, 4 512
486 MULI 0, 0 513
487 OFFSET 0, 0 514
488 LITI 0, 3 515
489 STX 0, 0 516
490 POP 0, 1 517
491 LDA 1, 24 518
492 LITI 0, 9 519
493 LITI 0, 4 520
494 MULI 0, 0 521
495 OFFSET 0, 0 522
496 LITI 0, 3 523
497 STX 0, 0 524
498 POP 0, 1 525
499 LDA 1, 24 526
500 LITI 0, 9 527
501 LITI 0, 4 528
502 MULI 0, 0 529
503 OFFSET 0, 0 530
504 LITI 0, 3 531
505 STX 0, 0 532
506 POP 0, 1 533
507 LDA 1, 24 534
508 LITI 0, 9 535
509 LITI 0, 4 536
510 MULI 0, 0 537
511 OFFSET 0, 0 538
512 LITI 0, 3 539
513 STX 0, 0 540
514 POP 0, 1 541
515 LDA 1, 24 542
516 LITI 0, 9 543
517 LITI 0, 4 544
518 MULI 0, 0 545
519 OFFSET 0, 0 546
520 LITI 0, 3 547
521 STX 0, 0 548
522 POP 0, 1 549
523 LDA 1, 24 550
524 LITI 0, 9 551
525 LITI 0, 4 552
526 MULI 0, 0 553
527 OFFSET 0, 0 554
528 LITI 0, 3 555
529 STX 0, 0 556
530 POP 0, 1 557
531 LDA 1, 24 558
532 LITI 0, 9 559
533 LITI 0, 4 560
534 MULI 0, 0 561
535 OFFSET 0, 0 562
536 LITI 0, 3 563
537 STX 0, 0 564
538 POP 0, 1 565
539 LDA 1, 24 566
540 LITI 0, 9 567
541 LITI 0, 4 568
542 MULI 0, 0 569
543 OFFSET 0, 0 570
544 LITI 0, 3 571
545 STX 0, 0 572
546 POP 0, 1 573
547 LDA 1, 24 574
548 LITI 0, 9 575
549 LITI 0, 4 576
550 MULI 0, 0 577
551 OFFSET 0, 0 578
552 LITI 0, 3 579
553 STX 0, 0 580
554 POP 0, 1 581
555 LDA 1, 24 582
556 LITI 0, 9 583
557 LITI 0, 4 584
558 MULI 0, 0 585
559 OFFSET 0, 0 586
560 LITI 0, 3 587
561 STX 0, 0 588
562 POP 0, 1 589
563 LDA 1, 24 590
564 LITI 0, 9 591
565 LITI 0, 4 592
566 MULI 0, 0 593
567 OFFSET 0, 0 594
568 LITI 0, 3 595
569 STX 0, 0 596
570 POP 0, 1 597
571 LDA 1, 24 598
572 LITI 0, 9 599
573 LITI 0, 4 600
574 MULI 0, 0 601
575 OFFSET 0, 0 602
576 LITI 0, 3 603
577 STX 0, 0 604
578 POP 0, 1 605
579 LDA 1, 24 606
580 LITI 0, 9 607
581 LITI 0, 4 608
582 MULI 0, 0 609
583 OFFSET 0, 0 610
584 LITI 0, 3 611
585 STX 0, 0 612
586 POP 0, 1 613
587 LDA 1, 24 614
588 LITI 0, 9 615
589 LITI 0, 4 616
590 MULI 0, 0 617
591 OFFSET 0, 0 618
592 LITI 0, 3 619
593 STX 0, 0 620
594 POP 0, 1 621
595 LDA 1, 24 622
596 LITI 0, 9 623
597 LITI 0, 4 624
598 MULI 0, 0 625
599 OFFSET 0, 0 626
600 LITI 0, 3 627
601 STX 0, 0 628
602 POP 0, 1 629
603 LDA 1, 24 630
604 LITI 0, 9 631
605 LITI 0, 4 632
606 MULI 0, 0 633
607 OFFSET 0, 0 634
608 LITI 0, 3 635
609 STX 0, 0 636
610 POP 0, 1 637
611 LDA 1, 24 638
612 LITI 0, 9 639
613 LITI 0, 4 640
614 MULI 0, 0 641
615 OFFSET 0, 0 642
616 LITI 0, 3 643
617 STX 0, 0 644
618 POP 0, 1 645
619 LDA 1, 24 646
620 LITI 0, 9 647
621 LITI 0, 4 648
622 MULI 0, 0 649
623 OFFSET 0, 0 650
624 LITI 0, 3 651
625 STX 0, 0 652
626 POP 0, 1 653
627 LDA 1, 24 654
628 LITI 0, 9 655
629 LITI 0, 4 656
630 MULI 0, 0 657
631 OFFSET 0, 0 658
632 LITI 0, 3 659
633 STX 0, 0 660
634 POP 0, 1 661
635 LDA 1, 24 662
636 LITI 0, 9 663
637 LITI 0, 4 664
638 MULI 0, 0 665
639 OFFSET 0, 0 666
640 LITI 0, 3 667
641 STX 0, 0 668
642 POP 0, 1 669
643 LDA 1, 24 670
644 LITI 0, 9 671
645 LITI 0, 4 672
646 MULI 0, 0 673
647 OFFSET 0, 0 674
648 LITI 0, 3 675
649 STX 0, 0 676
650 POP 0, 1 677
651 LDA 1, 24 678
652 LITI 0, 9 679
653 LITI 0, 4 680
654 MULI 0, 0 681
655 OFFSET 0, 0 682
656 LITI 0, 3 683
657 STX 0, 0 684
658 POP 0, 1 685
659 LDA 1, 24 686
660 LITI 0, 9 687
661 LITI 0, 4 688
662 MULI 0, 0 689
663 OFFSET 0, 0 690
664 LITI 0, 3 691
665 STX 0, 0 692
666 POP 0, 1 693
667 LDA 1, 24 694
668 LITI 0, 9 695
669 LITI 0, 4 696
670 MULI 0, 0 697
671 OFFSET 0, 0 698
672 LITI 0, 3 699
673 STX 0, 0 700
674 POP 0, 1 701
675 LDA 1, 24 702
676 LITI 0, 9 703
677 LITI 0, 4 704
678 MULI 0, 0 705
679 OFFSET 0, 0 706
680 LITI 0, 3 707
681 STX 0, 0 708
682 POP 0, 1 709
683 LDA 1, 24 710
684 LITI 0, 9 711
685 LITI 0, 4 712
686 MULI 0, 0 713
687 OFFSET 0, 0 714
688 LITI 0, 3 715
689 STX 0, 0 716
690 POP 0, 1 717
691 LDA 1, 24 718
692 LITI 0, 9 719
693 LITI 0, 4 720
694 MULI 0, 0 721
695 OFFSET 0, 0 722
696 LITI 0, 3 723
697 STX 0, 0 724
698 POP 0, 1 725
699 LDA 1, 24 726
700 LITI 0, 9 727
701 LITI 0, 4 728
702 MULI 0, 0 729
703 OFFSET 0, 0 730
704 LITI 0, 3 731
705 STX 0, 0 732
706 POP 0, 1 733
707 LDA 1, 24 734
708 LITI 0, 9 735
709 LITI 0, 4 736
710 MULI 0, 0 737
711 OFFSET 0, 0 738
712 LITI 0, 3 739
713 STX 0, 0 740
714 POP 0, 1 741
715 LDA 1, 24 742
716 LITI 0, 9 743
717 LITI 0, 4 744
718 MULI 0, 0 745
719 OFFSET 0, 0 746
720 LITI 0, 3 747
721 STX 0, 0 748
722 POP 0, 1 749
723 LDA 1, 24 750
724 LITI 0, 9 751
725 LITI 0, 4 752
726 MULI 0, 0 753
727 OFFSET 0, 0 754
728 LITI 0, 3 755
729 STX 0, 0 756
730 POP 0, 1 757
731 LDA 1, 24 758
732 LITI 0, 9 759
733 LITI 0, 4 760
734 MULI 0, 0 761
735 OFFSET 0, 0 762
736 LITI 0, 3 763
737 STX 0, 0 764
738 POP 0, 1 765
739 LDA 1, 24 766
740 LITI 0, 9 767
741 LITI 0, 4 768
742 MULI 0, 0 769
743 OFFSET 0, 0 770
744 LITI 0, 3 771
745 STX 0, 0 772
746 POP 0, 1 773
747 LDA 1, 24 774
748 LITI 0, 9 775
749 LITI 0, 4 776
750 MULI 0, 0 777
751 OFFSET 0, 0 778
752 LITI 0, 3 779
753 STX 0, 0 780
754 POP 0, 1 781
755 LDA 1, 24 782
756 LITI 0, 9 783
757 LITI 0, 4 784
758 MULI 0, 0 785
759 OFFSET 0, 0 786
760 LITI 0, 3 787
761 STX 0, 0 788
762 POP 0, 1 789
763 LDA 1, 24 790
764 LITI 0, 9 791
765 LITI 0, 4 792
766 MULI 0, 0 793
767 OFFSET 0, 0 794
768 LITI 0, 3 795
769 STX 0, 0 796
770 POP 0, 1 797
771 LDA 1, 24 798
772 LITI 0, 9 799
773 LITI 0, 4 800
774 MULI 0, 0 801
775 OFFSET 0, 0 802
776 LITI 0, 3 803
777 STX 0, 0 804
778 POP 0, 1 805
779 LDA 1, 24 806
780 LITI 0, 9 807
781 LITI 0, 4 808
782 MULI 0, 0 809
783 OFFSET 0, 0 810
784 LITI 0, 3 811
785 STX 0, 0 812
786 POP 0, 1 813
787 LDA 1, 24 814
788 LITI 0, 9 815
789 LITI 0, 4 816
790 MULI 0, 0 817
791 OFFSET 0, 0 818
792 LITI 0, 3 819
793 STX 0, 0 820
794 POP 0, 1 821
795 LDA 1, 24 822
796 LITI 0, 9 823
797 LITI 0, 4 824
798 MULI 0, 0 825
799 OFFSET 0, 0 826
800 LITI 0, 3 827
801 STX 0, 0 828
802 POP 0, 1 829
803 LDA 1, 24 830
804 LITI 0, 9 831
805 LITI 0, 4 832
806 MULI 0, 0 833
807 OFFSET 0, 0 834
808 LITI 0, 3 835
809 STX 0, 0 836
810 POP 0, 1 837
811 LDA 1, 24 838
812 LITI 0, 9 
```



```

85      POP      0, 1
86      LDA      1, 12
87      LITI     0, 0
88      STX      0, 0
89      POP      0, 1
90 L2:   LOD      1, 12
91      LITI     0, 10
92      LSSI     0, 0
93      JPC      0, L3
94      LDA      1, 64
95      LOD      1, 12
96      LITI     0, 4
97      MULI     0, 0
98      OFFSET   0, 0
99      LITI     0, 0
100     STX      0, 0
101     POP      0, 1
102
103 L1:   LOD      1, 12
104     LDA      1, 12
105     LDX      0, 0
106     INCI     0, 0
107     STO      0, 0
108     POP      0, 1
109     JMP      0, L2
110
111 L3:   LDA      1, 12
112     LITI     0, 0
113     STX      0, 0
114     POP      0, 1
115
116 L5:   LOD      1, 12
117     LITI     0, 10
118     LSSI     0, 0
119     JPC      0, L6
120     LDA      1, 16
121     LITI     0, 0
122     STX      0, 0
123     POP      0, 1
124
125 L8:   LOD      1, 16
126     LITI     0, 40
127     LITI     0, 4
128     DIVI     0, 0
129     LEQI     0, 0
130     JPC      0, L9
131     LOD      1, 12
132     LITI     0, 1
133
134     ADDI     0, 0
135     LDA      1, 24
136     LOD      1, 16
137     LITI     0, 4
138     MULI     0, 0
139     OFFSET   0, 0
140     LDI      0, 0
141     EQLI     0, 0
142     JPC      0, L10
143     LDA      1, 64
144     LOD      1, 12
145     LITI     0, 4
146     MULI     0, 0
147     OFFSET   0, 0
148     LDI      0, 0
149     LDA      1, 64
150     LOD      1, 12
151     LITI     0, 4
152     MULI     0, 0
153     OFFSET   0, 0
154     LDX      0, 0
155     INCI     0, 0
156     STO      0, 0
157     POP      0, 1
158 L10:  LOD      1, 16
159 L7:   LDA      1, 16
160     LDX      0, 0
161     INCI     0, 0
162     STO      0, 0
163     POP      0, 1
164     JMP      0, L8
165
166 L9:   LOD      1, 12
167     LDA      1, 12
168     LDX      0, 0
169     INCI     0, 0
170     STO      0, 0
171     POP      0, 1
172     JMP      0, L5
173
174 L6:   LDA      1, 20
175     LITI     0, 0
176     STX      0, 0
177     POP      0, 1
178
179 L12:  LOD      1, 20
180     LITI     0, 10
181     LSSI     0, 0
182
183     JPC      0, L10
184     INT      0, 12
185     LDA      0, 12
186     LOD      1, 20
187     LITI     0, 1
188     ADDI     0, 0
189     LDA      1, 64
190     LOD      1, 20
191     LITI     0, 4
192     MULI     0, 0
193     OFFSET   0, 0
194     LDI      0, 0
195     POP      0, 6
196     ADDR     0, printf
197     CAL      0, 0
198
199 L11:  LOD      1, 20
200     LDA      1, 20
201     LDX      0, 0
202     INCI     0, 0
203     STO      0, 0
204     POP      0, 1
205     JMP      0, L12
206
207 L13:  LDA      1, -4
208     LITI     0, 0
209     STO      0, 0
210     RET      0, 0
211
212 .literal 12 "The number of %d is %d\n"
213

```

● 어셈블리 프로그램을 시뮬레이터를 이용하여 실행하면 다음과 같은 결과가 나온다.

```

===== symbol =====
1: printf -1
2: malloc -2
3: scanf -3
4: main 3
5: L2 88
6: L3 107
7: L1 100
8: L5 111
9: L6 165
10: L8 119
11: L9 158
12: L10 151
13: L7 151
14: L4 158
15: L12 169
16: L13 194
17: L11 187
===== code =====
0: INT 0,40
1: SUP 0,3
2: RET 0,0
3: INT 0,104
4: LDA 1,24
5: LITI 0,0
6: LITI 0,4
7: MULI 0,0
8: OFFSET 0,0
9: LITI 0,2
10: STX 0,0
11: POP 0,1
12: LDA 1,24
13: LITI 0,1
14: LITI 0,4
15: MULI 0,0
16: OFFSET 0,0
17: LITI 0,8
18: STX 0,0

185: ADDR 0,-1
186: CAL 0,0
187: LOD 1,20
188: LDA 1,20
189: LDX 0,0
190: INCI 0,0
191: STO 0,0
192: POP 0,1
193: JMP 0,169
194: LDA 1,-4
195: LITI 0,0
196: STO 0,0
197: RET 0,0

start execution
The number of 1 is 2
The number of 2 is 3
The number of 3 is 3
The number of 4 is 0
The number of 5 is 1
The number of 6 is 0
The number of 7 is 0
The number of 8 is 2
The number of 9 is 1
The number of 10 is 0
end execution
minjeong@minjeong-VirtualBox: ~/m

```

(생략)

test9.c를 gcc로 실행하면 다음과 같은 결과가 나온다. 두 개가 동일한 결과라는 것을 확인할 수 있다.

```
minjeong@minjeong-VirtualBox:~/njeong/hw_final$ gcc -o gcc.exe test9.c
test9.c: In function 'main':
test9.c:28:3: warning: implicit declaration of function 'printf' [-Wimplicit-function-declaration]
   printf("The number of %d is %d\n", k + 1, count[k]);
   ^
test9.c:28:3: warning: incompatible implicit declaration of built-in function 'printf'
test9.c:28:3: note: include '<stdio.h>' or provide a declaration of 'printf'
minjeong@minjeong-VirtualBox:~/njeong/hw_final$ ./gcc.exe
The number of 1 is 2
The number of 2 is 3
The number of 3 is 3
The number of 4 is 0
The number of 5 is 1
The number of 6 is 0
The number of 7 is 0
The number of 8 is 2
The number of 9 is 1
The number of 10 is 0
minjeong@minjeong-VirtualBox:~/njeong/hw_final$
```



10) test10.c

배열안의 값을 오름차순으로 정렬하는 프로그램이다.

```
test10.c (~/mjeong/hw_final) - VIM
1 void sort(int a[], int n){
2     int i,j,x;
3     for(i=0;i<n;i++)
4         for(j=i+1;j<n;j=j+1)
5             if(a[i] > a[j]){
6                 x=a[i];
7                 a[i]=a[j];
8                 a[j]=x;
9             }
10 }
11
12 void main(){
13
14     int arr[7], i;
15     arr[0] = 4;
16     arr[1] = 6;
17     arr[2] = 2;
18     arr[3] = 7;
19     arr[4] = 9;
20     arr[5] = 11;
21     arr[6] = 1;
22     for(i=0;i<7;i++){
23         printf("%d ",arr[i]);
24     }
25     printf("\n sort()\n");
26     sort(arr,7);
27
28     for(i=0;i<7;i++){
29         printf("%d ",arr[i]);
30     }
31     printf("\n");
32 }
```

● 컴파일러를 통하여 만들어진 어셈블러 프로그램은 다음과 같다.

```
a.asm (~/mjeong/hw_final) - VIM
1  INT 0, 52
2  SUP 0, main
3  RET 0, 0
4  sort:
5      INT 0, 32
6      LDA 1, 20
7      LITI 0, 0
8      STX 0, 0
9      POP 0, 1
10 L2:
11     LOD 1, 20
12     LOD 1, 16
13     LSSI 0, 0
14     JPC 0, L3
15     LDA 1, 24
16     LOD 1, 20
17     LITI 0, 1
18     ADDI 0, 0
19     STX 0, 0
20     POP 0, 1
21 L5:
22     LOD 1, 24
23     LOD 1, 16
24     LSSI 0, 0
25     JPC 0, L6
26     LOD 1, 12
27     LOD 1, 20
28     LITI 0, 4
29     MULI 0, 0
30     OFFSET 0, 0
31     LDI 0, 0
32     LOD 1, 12
33     LOD 1, 24
34     LITI 0, 4
35     MULI 0, 0
36     OFFSET 0, 0
37     LDI 0, 0
38     GTRI 0, 0
39     JPC 0, L7
40     LDA 1, 28
41     LOD 1, 12
42     LOD 1, 20
43     LITI 0, 0
44     STX 0, 0
45     POP 0, 1
46     JPC 0, L8
47     LDA 1, 40
48     LITI 0, 7
49     LSSI 0, 0
50     JPC 0, L9
51     INT 0, 12
52     LDA 0, 36
53     LDA 1, 12
54     LOD 1, 40
55     LITI 0, 4
56     MULI 0, 0
57     OFFSET 0, 0
58     LDI 0, 0
59     POP 0, 5
60     ADDR 0, printf
61     CAL 0, 0
62     L11:
63     LOD 1, 40
64     LDA 1, 40
65     LDX 0, 0
66     INCI 0, 0
67     STO 0, 0
68     POP 0, 1
69     JMP 0, L12
70 L13:
71     INT 0, 12
72     LDA 0, 44
73     POP 0, 4
74     ADDR 0, printf
75     CAL 0, 0
76     RET 0, 0
77 .literal 12 "%d "
78 .literal 20 "\n sort()\n"
79 .literal 36 "%d "
80 .literal 44 "\n"
186 ADDR 0, sort
187 CAL 0, 0
188 LDA 1, 40
189 LITI 0, 0
190 STX 0, 0
191 POP 0, 1
192 L12:
193     LOD 1, 40
194     LITI 0, 7
195     LSSI 0, 0
196     JPC 0, L13
197     INT 0, 12
198     LDA 0, 36
199     LDA 1, 12
200     LOD 1, 40
201     LITI 0, 4
202     MULI 0, 0
203     OFFSET 0, 0
204     LDI 0, 0
205     POP 0, 5
206     ADDR 0, printf
207     CAL 0, 0
208 L11:
209     LOD 1, 40
210     LDA 1, 40
211     LDX 0, 0
212     INCI 0, 0
213     STO 0, 0
214     POP 0, 1
215     JMP 0, L12
216 L13:
217     INT 0, 12
218     LDA 0, 44
219     POP 0, 4
220     ADDR 0, printf
221     CAL 0, 0
222     RET 0, 0
223 .literal 12 "%d "
224 .literal 20 "\n sort()\n"
225 .literal 36 "%d "
226 .literal 44 "\n"
```

(생략)

- 어셈블리 프로그램을 시뮬레이터를 이용하여 실행하면 다음과 같은 결과가 나온다.

```

===== symbol =====
1: printf -1
2: malloc -2
3: scanf -3
4: main 81
5: sort 3
6: L2 8
7: L3 80
8: L5 18
9: L6 73
10: L7 66
11: L4 66
12: L1 73
13: L9 142
14: L10 164
15: L8 157
16: L12 179
17: L13 201
18: L11 194
===== code =====
0: INT 0,52
1: SUP 0,81
2: RET 0,0
3: INT 0,32
4: LDA 1,20
5: LITI 0,0
6: STX 0,0
7: POP 0,1
8: LOD 1,20
9: LOD 1,16
10: LSSI 0,0
11: JPC 0,80
12: LDA 1,24
13: LOD 1,20
14: LITI 0,1
15: ADDI 0,0
16: STX 0,0
17: POP 0,1
18: LOD 1,24
19: LOD 1,16
20: LSSI 0,0
21: JPC 0,73
22: LOD 1,12
23: LOD 1,20

179: LOD 1,40
180: LITI 0,7
181: LSSI 0,0
182: JPC 0,201
183: INT 0,12
184: LDA 0,36
185: LDA 1,12
186: LOD 1,40
187: LITI 0,4
188: MULI 0,0
189: OFFSET 0,0
190: LDI 0,0
191: POP 0,5
192: ADDR 0,-1
193: CAL 0,0
194: LOD 1,40
195: LDA 1,40
196: LDX 0,0
197: INCI 0,0
198: STO 0,0
199: POP 0,1
200: JMP 0,179
201: INT 0,12
202: LDA 0,44
203: POP 0,4
204: ADDR 0,-1
205: CAL 0,0
206: RET 0,0

start execution
4 6 2 7 9 11 1
sort()
1 2 4 6 7 9 11
end execution
minjeong@minjeong-Virtua

```

(생략)

test10.c를 gcc로 실행하면 다음과 같은 결과가 나온다. 두 개가 동일한 결과라는 것을 확인할 수 있다.

```

minjeong@minjeong-VirtualBox:~/mjeong/hw_final$ gcc -o gcc.exe test10.c
test10.c: In function 'main':
test10.c:23:3: warning: implicit declaration of function 'printf' [-Wimplicit-function-decl
aration]
    printf("%d ",arr[i]);
    ^
test10.c:23:3: warning: incompatible implicit declaration of built-in function 'printf'
test10.c:23:3: note: include '<stdio.h>' or provide a declaration of 'printf'
test10.c:25:2: warning: incompatible implicit declaration of built-in function 'printf'
    printf("\n sort()\n");
    ^
test10.c:25:2: note: include '<stdio.h>' or provide a declaration of 'printf'
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$ ./gcc.exe
4 6 2 7 9 11 1
sort()
1 2 4 6 7 9 11
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$

```

11) test11.c

```
test11.c (~/mjeong/hw_final) - VIM
1 void func2()
2 {
3     printf("func2() is running...\n");
4     return;
5 }
6 void func3()
7 {
8     printf("func3() is running...\n");
9     return;
10 }
11 void func4()
12 {
13     printf("func4() is running...\n");
14     return;
15 }
16 void func1()
17 {
18     printf("func1() is running...\n");
19     func3();
20     return;
21 }
22
23 int main()
24 {
25     func2();
26     func1();
27     return 0;
28 }
29
30
```

- 컴파일러를 통하여 만들어진 어셈블러 프로그램은 다음과 같다.

```
a.asm (~/mjeong/hw_final) - VIM
1      INT    0, 124
2      SUP    0, main
3      RET    0, 0
4 func2:
5      INT    0, 12
6      INT    0, 12
7      LDA    0, 12
8      POP    0, 4
9      ADDR   0, printf
10     CAL    0, 0
11     RET    0, 0
12 func3:
13     INT    0, 12
14     INT    0, 12
15     LDA    0, 40
16     POP    0, 4
17     ADDR   0, printf
18     CAL    0, 0
19     RET    0, 0
20 func4:
21     INT    0, 12
22     INT    0, 12
23     LDA    0, 68
24     POP    0, 4
25     ADDR   0, printf
26     CAL    0, 0
27     RET    0, 0
28 func1:
29     INT    0, 12
30     INT    0, 12
31     LDA    0, 96
32     POP    0, 4
33     ADDR   0, printf
34     CAL    0, 0
35     INT    0, 12
36     POP    0, 3
37     ADDR   0, func3
38     CAL    0, 0
39     RET    0, 0
40 main:
41     INT    0, 12
42     INT    0, 12
43     POP    0, 3
44     ADDR   0, func2
45     CAL    0, 0
46     INT    0, 12
47     POP    0, 3
48     ADDR   0, func1
49     CAL    0, 0
50     LDA    1, -4
51     LITI   0, 0
52     STO    0, 0
53     RET    0, 0
54 .literal 12 "func2() is running...\n"
55 .literal 40 "func3() is running...\n"
56 .literal 68 "func4() is running...\n"
57 .literal 96 "func1() is running...\n"
```

- 어셈블러 프로그램을 시뮬레이터를 이용하여 실행하면 다음과 같은 결과가 나온다.



```

===== symbol =====
1: printf -1
2: malloc -2
3: scanf -3
4: main 35
5: func2 3
6: func3 10
7: func4 17
8: func1 24
===== code =====
0: INT 0,124
1: SUP 0,35
2: RET 0,0
3: INT 0,12
4: INT 0,12
5: LDA 0,12
6: POP 0,4
7: ADDR 0,-1
8: CAL 0,0
9: RET 0,0
10: INT 0,12
11: INT 0,12
12: LDA 0,40
13: POP 0,4
14: ADDR 0,-1
15: CAL 0,0
16: RET 0,0
17: INT 0,12
18: INT 0,12
19: LDA 0,68
20: POP 0,4
21: ADDR 0,-1
22: CAL 0,0
23: RET 0,0
24: INT 0,12
25: INT 0,12
26: LDA 0,96
27: POP 0,4
28: ADDR 0,-1
29: CAL 0,0
30: INT 0,12
31: POP 0,3
32: ADDR 0,10
33: CAL 0,0
34: RET 0,0
35: INT 0,12
36: INT 0,12
37: POP 0,3
38: ADDR 0,3
39: CAL 0,0
40: INT 0,12
41: POP 0,3
42: ADDR 0,24
43: CAL 0,0
44: LDA 1,-4
45: LITI 0,0
46: STO 0,0
47: RET 0,0
start execution
func2() is running...
func1() is running...
func3() is running...
end execution
minjeong@minjeong-Virtua

```

test11.c를 gcc로 실행하면 다음과 같은 결과가 나온다. 두 개가 동일한 결과라는 것을 확인할 수 있다.

```

minjeong@minjeong-VirtualBox:~/mjeong/hw_final$ gcc -o gcc.exe test11.c
test11.c: In function 'func2':
test11.c:3:2: warning: implicit declaration of function 'printf' [-Wimplicit-function-declaration]
    printf("func2() is running...\n");
    ^
test11.c:3:2: warning: incompatible implicit declaration of built-in function 'printf'
test11.c:3:2: note: include '<stdio.h>' or provide a declaration of 'printf'
test11.c: In function 'func3':
test11.c:8:2: warning: incompatible implicit declaration of built-in function 'printf'
    printf("func3() is running...\n");
    ^
test11.c:8:2: note: include '<stdio.h>' or provide a declaration of 'printf'
test11.c: In function 'func4':
test11.c:13:2: warning: incompatible implicit declaration of built-in function 'printf'
    printf("func4() is running...\n");
    ^
test11.c:13:2: note: include '<stdio.h>' or provide a declaration of 'printf'
test11.c: In function 'func1':
test11.c:18:2: warning: incompatible implicit declaration of built-in function 'printf'
    printf("func1() is running...\n");
    ^
test11.c:18:2: note: include '<stdio.h>' or provide a declaration of 'printf'
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$ ./gcc.exe
func2() is running...
func1() is running...
func3() is running...
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$

```

12) test12.c

소수를 구하는 프로그램이다.

```
test12.c (~/mjeong/hw_final) - VIM
1 void main(){
2     int n,x,i,k,lim,prim;
3     int p[300];
4     p[1]=2;
5     printf("%d ",2);
6     n=100;
7     x=1;
8     lim=1;
9     for(i=2;i<=n;i++){
10        do{
11            x=x+1;x++;
12            if(p[lim]*p[lim] <=x)
13                lim++;
14            k=2;
15            prim=1;
16            while(prim && (k<lim)){
17                prim=x%p[k];
18                k++;}
19            while(!prim);
20            p[i]=x;
21            printf("%d ",x);
22            if(i%10==0) printf("\n");
23        }
24    }
```

● 컴파일러를 통하여 만들어진 어셈블러 프로그램은 다음과 같다.

```
a.asm (~/mjeong/hw_final) - VIM
1  INT    0, 36
2  SUP    0, main
3  RET     0, 0
4  main:
5      INT    0, 1236
6      LDA    1, 36
7      LITI   0, 1
8      LITI   0, 4
9      MULI   0, 0
10     OFFSET 0, 0
11     LITI   0, 2
12     STX    0, 0
13     POP     0, 1
14     INT     0, 12
15     LDA     0, 12
16     LITI   0, 2
17     POP     0, 5
18     ADDR    0, printf
19     CAL     0, 0
20     LDA     1, 12
21     LITI   0, 100
22     STX     0, 0
23     POP     0, 1
24     LDA     1, 16
25     LITI   0, 1
26     STX     0, 0
27     POP     0, 1
28     LDA     1, 28
29     LITI   0, 1
30     STX     0, 0
31     POP     0, 1
32     LDA     1, 20
33     LITI   0, 2
34     STX     0, 0
35     POP     0, 1
36 L2:
37     LOD     1, 20
38     LOD     1, 12
39     LEQI    0, 0
40     JPC     0, L3
41 L6:
42     LDA     1, 16
43     LOD     1, 16
44     LITI   0, 1
45     ADDI    0, 0
46     STX     0, 0
47     POP     0, 1
48     LOD     1, 16
49     LDA     1, 16
50     LDX     0, 0
51     INCI    0, 0
52     STO     0, 0
53     POP     0, 1
54     LDA     1, 36
55     LOD     1, 28
56     LITI   0, 4
57     MULI   0, 0
58     OFFSET 0, 0
59     LDI     0, 0
60     LDA     1, 36
61     LOD     1, 28
62     LITI   0, 4
63     MULI   0, 0
64     OFFSET 0, 0
65     LDI     0, 0
66     MULI   0, 0
67     LOD     1, 16
68     LEQI    0, 0
69     JPC     0, L7
70     LOD     1, 28
71     LDA     1, 28
72     LDX     0, 0
73     INCI    0, 0
74     STO     0, 0
75     POP     0, 1
76 L7:
77     LDA     1, 24
78     LITI   0, 2
79     STX     0, 0
80     POP     0, 1
```



```

81      LDA    1, 32
82      LITI   0, 1
83      STX    0, 0
84      POP    0, 1
85 L9:
86      LOD    1, 32
87      JPCR   0, L10
88      LOD    1, 24
89      LOD    1, 28
90      LSSI   0, 0
91 L10:
92      JPC    0, L11
93      LDA    1, 32
94      LOD    1, 16
95      LDA    1, 36
96      LOD    1, 24
97      LITI   0, 4
98      MULI   0, 0
99      OFFSET 0, 0
100     LDI     0, 0
101     MOD     0, 0
102     STX     0, 0
103     POP     0, 1
104     LOD     1, 24
105     LDA     1, 24
106     LDX     0, 0
107     INCI    0, 0
108     STO     0, 0
109     POP     0, 1
110 L8:
111     JMP     0, L9
112 L11:
113 L5:
114     LOD     1, 32
115     NOT     0, 0
116     JPT     0, L6
117 L4:
118     LDA     1, 36
119     LOD     1, 20
120     LITI    0, 4
121     MULI    0, 0
122     OFFSET  0, 0
123     LOD     1, 16
124     STX     0, 0
125     POP     0, 1
126     INT     0, 12
127     LDA     0, 20
128     LOD     1, 16
129     POP     0, 5
130     ADDR    0, printf
131     CAL     0, 0
132     LOD     1, 20
133     LITI    0, 10
134     MOD     0, 0
135     LITI    0, 0
136     EQLI    0, 0
137     JPC     0, L12
138     INT     0, 12
139     LDA     0, 28
140     POP     0, 4
141     ADDR    0, printf
142     CAL     0, 0
143 L12:
144 L1:
145     LOD     1, 20
146     LDA     1, 20
147     LDX     0, 0
148     INCI    0, 0
149     STO     0, 0
150     POP     0, 1
151     JMP     0, L2
152 L3:
153     RET     0, 0
154 .literal   12  "%d "
155 .literal   20  "%d "
156 .literal   28  "\n"
~

```

- 어셈블러 프로그램을 시뮬레이터를 이용하여 실행하면 다음과 같은 결과가 나온다.

```

===== symbol =====
1:  printf    -1
2:  malloc    -2
3:  scanf    -3
4:  main      3
5:  L2        34
6:  L3        139
7:  L6        38
8:  L7        72
9:  L9        80
10: L10       85
11: L11       104
12: L8        103
13: L5        104
14: L4        107
15: L12       132
16: L1        132
===== code =====
0:  INT       0,36
1:  SUP       0,3
2:  RET       0,0
3:  INT       0,1236
4:  LDA       1,36
5:  LITI      0,1
6:  LITI      0,4
7:  MULI      0,0
8:  OFFSET    0,0
9:  LITI      0,2
10: STX       0,0
11: POP       0,1
12: INT       0,12
13: LDA       0,12
14: LITI      0,2
15: POP       0,5
16: ADDR      0,-1

```

(생략)

```

117: LOD      1,16
118: POP       0,5
119: ADDR      0,-1
120: CAL       0,0
121: LOD      1,20
122: LITI      0,10
123: MOD       0,0
124: LITI      0,0
125: EQLI      0,0
126: JPC       0,132
127: INT       0,12
128: LDA       0,28
129: POP       0,4
130: ADDR      0,-1
131: CAL       0,0
132: LOD      1,20
133: LDA       1,20
134: LDX       0,0
135: INCI      0,0
136: STO       0,0
137: POP       0,1
138: JMP       0,34
139: RET       0,0
start execution
2 3 5 7 11 13 17 19 23 29
31 37 41 43 47 53 59 61 67 71
73 79 83 89 97 101 103 107 109 113
127 131 137 139 149 151 157 163 167 173
179 181 191 193 197 199 211 223 227 229
233 239 241 251 257 263 269 271 277 281
283 293 307 311 313 317 331 337 347 349
353 359 367 373 379 383 389 397 401 409
419 421 431 433 439 443 449 457 461 463
467 479 487 491 499 503 509 521 523 541
end execution

```

test12.c를 gcc로 실행하면 다음과 같은 결과가 나온다. 두 개가 동일한 결과라는 것을 확인할 수 있다.

```
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$ gcc -o gcc.exe test12.c
test12.c: In function 'main':
test12.c:5:2: warning: implicit declaration of function 'printf' [-Wimplicit-function-declaration]
    printf("%d ",2);
    ^
test12.c:5:2: warning: incompatible implicit declaration of built-in function 'printf'
test12.c:5:2: note: include '<stdio.h>' or provide a declaration of 'printf'
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$ ./gcc.exe
2 3 5 7 11 13 17 19 23 29
31 37 41 43 47 53 59 61 67 71
73 79 83 89 97 101 103 107 109 113
127 131 137 139 149 151 157 163 167 173
179 181 191 193 197 199 211 223 227 229
233 239 241 251 257 263 269 271 277 281
283 293 307 311 313 317 331 337 347 349
353 359 367 373 379 383 389 397 401 409
419 421 431 433 439 443 449 457 461 463
467 479 487 491 499 503 509 521 523 541
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$
```

13) test13.c

```

test13.c (~mjeong/hw_final) - VIM
1 int strcmp(char *s, char *t){
2     for( ;*s==*t;s++){
3         if(*s==0)
4             return 0;
5         t++;}
6     return *s-*t;
7 }
8
9 void result(char *s, char *t, int ret){
10     if(ret == 0)
11         printf("ret : %d\n %s and %s are the same\n",ret, s, t);
12     else
13         printf("ret : %d\n %s and %s are not the same\n",ret, s, t);
14 }
15
16 void main(){
17     char *a, *b, *c, *d;
18     int ret;
19     a="computer";
20     b="computer";
21
22     ret = strcmp(a,b);
23     result(a,b,ret);
24
25     c="os";
26     d="compiler";
27
28     ret = strcmp(c,d);
29     result(c,d,ret);
30 }

```

- 컴파일러를 통하여 만들어진 어셈블리 프로그램은 다음과 같다.

```

a.asm (~mjeong/hw_final) - V
1      INT    0, 140
2      SUP    0, main
3      RET    0, 0
4 strcmp:
5      INT    0, 20
6 L2:
7      LOD    1, 12
8      LDIB   0, 0
9      LOD    1, 16
10     LDIB   0, 0
11     EQLI   0, 0
12     JPC    0, L3
13     LOD    1, 12
14     LDIB   0, 0
15     LIYI   0, 0
16     EQLI   0, 0
17     JPC    0, L4
18     LDA    1, -4
19     LIYI   0, 0
20     STO    0, 0
21 L4:
22     LOD    1, 16
23     LDA    1, 16
24     LDX    0, 0
25     LIYI   0, 1
26     ADDI   0, 0
27     STO    0, 0
28     POP    0, 1
29 L1:
30     LOD    1, 12
31     LDA    1, 12
32     LDX    0, 0
33     LIYI   0, 1
34     ADDI   0, 0
35     STO    0, 0

```

(생략)

```

100     LDA    0, 120
101     STX    0, 0
102     POP    0, 1
103     LDA    1, 24
104     LDA    0, 128
105     STX    0, 0
106     POP    0, 1
107     LDA    1, 28
108     INT    0, 16
109     LOD    1, 20
110     LOD    1, 24
111     POP    0, 5
112     ADDR   0, strcmp
113     CAL    0, 0
114     STX    0, 0
115     POP    0, 1
116     INT    0, 1
117     LOD    1, 20
118     LOD    1, 24
119     LOD    1, 28
120     POP    0, 6
121     ADDR   0, result
122     CAL    0, 0
123     RET    0, 0
124 .literal 12 "ret : %d\n %s and %s are the same\n"
125 .literal 52 "ret : %d\n %s and %s are not the same\n"
126 .literal 96 "computer"
127 .literal 108 "computer"
128 .literal 120 "os"
129 .literal 128 "compiler"

```

- 어셈블리 프로그램을 시뮬레이터를 이용하여 실행하면 다음과 같은 결과가 나온다.

```

===== symbol =====
 1: printf  -1
 2: malloc  -2
 3: scanf   -3
 4: main    64
 5: strcmp  3
 6: L2      4
 7: L3      33
 8: L4      18
 9: L1      25
10: result  41
11: L5      55
12: L6      63
===== code =====
 0: INT     0,140
 1: SUP     0,64
 2: RET     0,0
 3: INT     0,20
 4: LOD     1,12
 5: LDIB    0,0
 6: LOD     1,16
 7: LDIB    0,0
 8: EQLI    0,0
 9: JPC     0,33
10: LOD     1,12
11: LDIB    0,0
12: LITI    0,0

```

(생략)

```

101: POP     0,5
102: ADDR    0,3
103: CAL     0,0
104: STX     0,0
105: POP     0,1
106: INT     0,12
107: LOD     1,20
108: LOD     1,24
109: LOD     1,28
110: POP     0,6
111: ADDR    0,41
112: CAL     0,0
113: RET     0,0
start execution
ret : -12
computer and computer are not the same
ret : 12
os and compiler are not the same
end execution
minjeong@minjeong-VirtualBox:~/minjeong/hw_final$

```

test13.c를 gcc로 실행하면 다음과 같은 결과가 나온다. 예상과 달리 시뮬레이터를 이용하여 실행한 결과가 틀린 결과가 나왔다. 포인터와 관련된 부분에서 뭔가 잘못 구현된 부분이 있으리라 추정된다.

```

minjeong@minjeong-VirtualBox:~/minjeong/hw_final$ gcc -o gcc.exe test13.c
test13.c: In function 'result':
test13.c:11:3: warning: implicit declaration of function 'printf' [-Wimplicit-function-declaration]
   printf("ret : %d\n %s and %s are the same\n",ret, s, t);
   ^
test13.c:11:3: warning: incompatible implicit declaration of built-in function 'printf'
test13.c:11:3: note: include '<stdio.h>' or provide a declaration of 'printf'
test13.c:13:3: warning: incompatible implicit declaration of built-in function 'printf'
   printf("ret : %d\n %s and %s are not the same\n",ret, s, t);
   ^
test13.c:13:3: note: include '<stdio.h>' or provide a declaration of 'printf'
minjeong@minjeong-VirtualBox:~/minjeong/hw_final$ ./gcc.exe
ret : 0
computer and computer are the same
ret : 12
os and compiler are not the same
minjeong@minjeong-VirtualBox:~/minjeong/hw_final$

```



14) test14.c

```
test14.c (~/mjeong/hw_final) - VIM
1 int gcd (int x, int y){
2     int a,b;
3     a=x;
4     b=y;
5     while(a != b){
6         if(a < b)
7             b=b-a;
8         if(a > b)
9             a=a-b;}
10    return (a);
11 }
12
13 void main(){
14     int i;
15     i=gcd(84,36);
16     printf("result=%d\n",i);
17 }
```

● 컴파일러를 통하여 만들어진 어셈블러 프로그램은 다음과 같다.

```
a.asm (~/mjeong/hw_final)
1      INT    0, 28
2      SUP    0, main
3      RET    0, 0
4 gcd:
5      INT    0, 28
6      LDA    1, 20
7      LOD    1, 12
8      STX    0, 0
9      POP    0, 1
10     LDA    1, 24
11     LOD    1, 16
12     STX    0, 0
13     POP    0, 1
14 L2:
15     LOD    1, 20
16     LOD    1, 24
17     NEQI    0, 0
18     JPC    0, L3
19     LOD    1, 20
20     LOD    1, 24
21     LSSI    0, 0
22     JPC    0, L4
23     LDA    1, 24
24     LOD    1, 24
25     LOD    1, 20
26     SUBI    0, 0
27     STX    0, 0
28     POP    0, 1
29 L4:
30     LOD    1, 20
31     LOD    1, 24
32     GTRI    0, 0
33     JPC    0, L5
34     LDA    1, 20
35     LOD    1, 20
36     LOD    1, 24
37     SUBI    0, 0
38     STX    0, 0
39     POP    0, 1
40 L5:
41 L1:
42     JMP    0, L2
43 L3:
44     LDA    1, -4
45     LOD    1, 20
46     STO    0, 0
47     RET    0, 0
48 main:
49     INT    0, 16
50     LDA    1, 12
51     INT    0, 16
52     LITI    0, 84
53     LITI    0, 36
54     POP    0, 5
55     ADDR    0, gcd
56     CAL    0, 0
57     STX    0, 0
58     POP    0, 1
59     INT    0, 12
60     LDA    0, 12
61     LOD    1, 12
62     POP    0, 5
63     ADDR    0, printf
64     CAL    0, 0
65     RET    0, 0
66 .literal 12 "result=%d\n"
```

- 어셈블리 프로그램을 시뮬레이터를 이용하여 실행하면 다음과 같은 결과가 나온다.

```

===== symbol =====
 1:  printf  -1
 2:  malloc  -2
 3:  scanf   -3
 4:  main    41
 5:  gcd      3
 6:  L2      12
 7:  L3      37
 8:  L4      26
 9:  L5      36
10:  L1      36
===== code =====
 0:  INT      0,28
 1:  SUP      0,41
 2:  RET      0,0
 3:  INT      0,28
 4:  LDA      1,20
 5:  LOD      1,12
 6:  STX      0,0
 7:  POP      0,1
 8:  LDA      1,24
 9:  LOD      1,16
10:  STX      0,0
11:  POP      0,1
12:  LOD      1,20
13:  LOD      1,24
14:  NEQI     0,0
15:  JPC      0,37
16:  LOD      1,20
17:  LOD      1,24
18:  LSSI     0,0
19:  JPC      0,26
20:  LDA      1,24
21:  LOD      1,24
22:  LOD      1,20
23:  SUBI     0,0
24:  STX      0,0
25:  POP      0,1
26:  LOD      1,20
27:  LOD      1,24
28:  GTRI     0,0
29:  JPC      0,36
30:  LDA      1,20
31:  LOD      1,20
32:  LOD      1,24
33:  SUBI     0,0
34:  STX      0,0
35:  POP      0,1
36:  JMP      0,12
37:  LDA      1,-4
38:  LOD      1,20
39:  STO      0,0
40:  RET      0,0
41:  INT      0,16
42:  LDA      1,12
43:  INT      0,16
44:  LITI     0,84
45:  LITI     0,36
46:  POP      0,5
47:  ADDR     0,3
48:  CAL      0,0
49:  STX      0,0
50:  POP      0,1
51:  INT      0,12
52:  LDA      0,12
53:  LOD      1,12
54:  POP      0,5
55:  ADDR     0,-1
56:  CAL      0,0
57:  RET      0,0
start execution
result=12
end execution
minjeong@minjeong-VirtualBox:~/

```

test14.c를 gcc로 실행하면 다음과 같은 결과가 나온다. 두 개가 동일한 결과라는 것을 확인할 수 있다.

```

minjeong@minjeong-VirtualBox:~/mjeong/hw_final$ gcc -o gcc.exe test14.c
test14.c: In function 'main':
test14.c:16:2: warning: implicit declaration of function 'printf' [-Wimplicit-function-declaration]
   printf("result=%d\n",i);
   ^
test14.c:16:2: warning: incompatible implicit declaration of built-in function 'printf'
test14.c:16:2: note: include '<stdio.h>' or provide a declaration of 'printf'
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$ ./gcc.exe
result=12
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$

```

15) test15.c

```

test15.c + (~/.mjeong/hw_final) - VIM
1 int mul(int a, int b){
2   int result;
3   result=0;
4   while(a){
5     if(a%2)
6       result=result+b;
7     a=a/2;
8     b=b*2;
9   }
10  return result;
11 }
12
13 void main(){
14   int i;
15   i=mul(120,3);
16   printf("result=%d\n",i);
17 }

```

- 컴파일러를 통하여 만들어진 어셈블러 프로그램은 다음과 같다.

```

a.asm (~/.mjeong/hw_final)
1      INT     0, 28
2      SUP     0, main
3      RET     0, 0
4 mul:
5      INT     0, 24
6      LDA     1, 20
7      LITI    0, 0
8      STX     0, 0
9      POP     0, 1
10 L2:
11      LOD     1, 12
12      JPC     0, L3
13      LOD     1, 12
14      LITI    0, 2
15      MOD     0, 0
16      JPC     0, L4
17      LDA     1, 20
18      LOD     1, 20
19      LOD     1, 16
20      ADDI    0, 0
21      STX     0, 0
22      POP     0, 1
23 L4:
24      LDA     1, 12
25      LOD     1, 12
26      LITI    0, 2
27      DIVI    0, 0
28      STX     0, 0
29      POP     0, 1
30      LDA     1, 16
31      LOD     1, 16
32      LITI    0, 2
33      MULI    0, 0
34      STX     0, 0
35      POP     0, 1
36 L1:
37      JMP     0, L2
38 L3:
39      LDA     1, -4
40      LOD     1, 20
41      STO     0, 0
42      RET     0, 0
43 main:
44      INT     0, 16
45      LDA     1, 12
46      INT     0, 16
47      LITI    0, 120
48      LITI    0, 3
49      POP     0, 5
50      ADDR    0, mul
51      CAL     0, 0
52      STX     0, 0
53      POP     0, 1
54      INT     0, 12
55      LDA     0, 12
56      LOD     1, 12
57      POP     0, 5
58      ADDR    0, printf
59      CAL     0, 0
60      RET     0, 0
61 .literal    12  "result=%d\n"

```



- 어셈블리 프로그램을 시뮬레이터를 이용하여 실행하면 다음과 같은 결과가 나온다.

```

===== symbol =====
 1: printf      -1
 2: malloc      -2
 3: scanf       -3
 4: main        37
 5: mul         3
 6: L2          8
 7: L3         33
 8: L4         20
 9: L1         32
===== code =====
 0: INT         0,28
 1: SUP         0,37
 2: RET         0,0
 3: INT         0,24
 4: LDA         1,20
 5: LITI        0,0
 6: STX         0,0
 7: POP         0,1
 8: LOD         1,12
 9: JPC         0,33
10: LOD         1,12
11: LITI        0,2
12: MOD         0,0
13: JPC         0,20
14: LDA         1,20
15: LOD         1,20
16: LOD         1,16
17: ADDI        0,0
18: STX         0,0
19: POP         0,1
20: LDA         1,12
21: LOD         1,12
22: LITI        0,2
23: DIVI        0,0
24: STX         0,0
25: POP         0,1
26: LDA         1,16
27: LOD         1,16
28: LITI        0,2
29: MULI        0,0
30: STX         0,0
31: POP         0,1
32: JMP         0,8
33: LDA         1,-4
34: LOD         1,20
35: STO         0,0
36: RET         0,0
37: INT         0,16
38: LDA         1,12
39: INT         0,16
40: LITI        0,120
41: LITI        0,3
42: POP         0,5
43: ADDR        0,3
44: CAL         0,0
45: STX         0,0
46: POP         0,1
47: INT         0,12
48: LDA         0,12
49: LOD         1,12
50: POP         0,5
51: ADDR        0,-1
52: CAL         0,0
53: RET         0,0
start execution
result=360
end execution
minjeong@minjeong-VirtualBox:

```

test15.c를 gcc로 실행하면 다음과 같은 결과가 나온다. 두 개가 동일한 결과라는 것을 확인할 수 있다.

```

minjeong@minjeong-VirtualBox:~/mjeong/hw_final$ gcc -o gcc.exe test15.c
test15.c: In function 'main':
test15.c:16:2: warning: implicit declaration of function 'printf' [-Wimplicit-function-declaration]
   printf("result=%d\n",i);
   ^
test15.c:16:2: warning: incompatible implicit declaration of built-in function 'printf'
test15.c:16:2: note: include '<stdio.h>' or provide a declaration of 'printf'
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$ ./gcc.exe
result=360
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$

```



16) test16.c

```

test16.c (~/.mjeong/hw_final) - VIM
1 int a[10];
2 void qsort(int l, int r){
3     int i,j,x,w,k;
4     i=l;
5     j=r;
6     x=a[(l+r)/2];
7     do{
8         while(a[i]<x)
9             i++;
10        while(x<a[j])
11            j--;
12        if(i<=j){
13            w=a[i];
14            a[i]=a[j];
15            a[j]=w;
16            i++;
17            j--;
18        }
19    }
20    while(i<=j);
21    if(l<j)
22        qsort(l,j);
23    if(i<r)
24        qsort(i,r);
25 }
26 void main(){
27     int k;
28     a[0]=0;a[1]=1;a[2]=3;a[3]=5;a[4]=7;
29     a[5]=9;a[6]=2;a[7]=4;a[8]=6;a[9]=8;
30     for(k=0;k<10;k++) printf("%d ",a[k]);printf("\n");
31     qsort(0,9);
32     for(k=0;k<10;k++) printf("%d ",a[k]);printf("\n");
33 }

```

● 컴파일러를 통하여 만들어진 어셈블러 프로그램은 다음과 같다.

<pre> a.asm (~/.mjeong/hw_final) - VIM 1  INT 0, 84 2  SUP 0, main 3  RET 0, 0 4  qsort: 5      INT 0, 40 6      LDA 1, 20 7      LOD 1, 12 8      STX 0, 0 9      POP 0, 1 10     LDA 1, 24 11     LOD 1, 16 12     STX 0, 0 13     POP 0, 1 14     LDA 1, 28 15     LOD 0, 12 16     LOD 1, 12 17     LOD 1, 16 18     ADDI 0, 2 19     LITI 0, 4 20     DIVI 0, 0 21     LITI 0, 4 22     MULI 0, 0 23     OFFSET 0, 0 24     LDI 0, 0 25     STX 0, 0 26     POP 0, 1 27 L3: 28 L5: 29     LDA 0, 12 30     LOD 1, 20 31     LITI 0, 4 32     MULI 0, 0 33     OFFSET 0, 0 34     LDI 0, 0 35     LOD 1, 28 36     LSSI 0, 0 37     JPC 0, L6 38     LOD 1, 20 39     LDA 1, 20 40     LDX 0, 0 41     INCI 0, 0 42     STO 0, 0 43     POP 0, 1 44 L4: 45     JMP 0, L5 46 L6: 47 L8: 48     LOD 1, 28 49     LDA 0, 12 50     LOD 1, 24 51     LITI 0, 4 52 </pre>	<pre> 52     MULI 0, 0 53     OFFSET 0, 0 54     LDI 0, 0 55     LSSI 0, 0 56     JPC 0, L9 57     LOD 1, 24 58     LDA 1, 24 59     LDX 0, 0 60     DECI 0, 0 61     STO 0, 0 62     POP 0, 1 63 L7: 64     JMP 0, L8 65 L9: 66     LOD 1, 20 67     LOD 1, 24 68     LEQI 0, 0 69     JPC 0, L10 70     LDA 1, 32 71     LDA 0, 12 72     LOD 1, 20 73     LITI 0, 4 74     MULI 0, 0 75     OFFSET 0, 0 76     LDI 0, 0 77     STX 0, 0 78     POP 0, 1 79     LDA 0, 12 80     LOD 1, 20 81     LITI 0, 4 82     MULI 0, 0 83     OFFSET 0, 0 84     LDA 0, 12 85     LOD 1, 24 86     LITI 0, 4 87     MULI 0, 0 88     OFFSET 0, 0 89     LDI 0, 0 90     STX 0, 0 91     POP 0, 1 92     LDA 0, 12 93     LOD 1, 24 94     LITI 0, 4 95     MULI 0, 0 96     OFFSET 0, 0 97     LOD 1, 32 98     STX 0, 0 99     POP 0, 1 100    LOD 1, 20 101    LDA 1, 20 102    LDX 0, 0 103    INCI 0, 0 </pre>	<pre> 251    JMP 0, L14 252 L15: 253    INT 0, 12 254    LDA 0, 60 255    POP 0, 4 256    ADDR 0, printf 257    CAL 0, 0 258    INT 0, 12 259    LITI 0, 0 260    LITI 0, 9 261    POP 0, 5 262    ADDR 0, qsort 263    CAL 0, 0 264    LDA 1, 12 265    LITI 0, 0 266    STX 0, 0 267    POP 0, 1 268 L17: 269    LOD 1, 12 270    LITI 0, 10 271    LSSI 0, 0 272    JPC 0, L18 273    INT 0, 12 274    LDA 0, 68 275    LDA 0, 12 276    LOD 1, 12 277    LITI 0, 4 278    MULI 0, 0 279    OFFSET 0, 0 280    LDI 0, 0 281    POP 0, 5 282    ADDR 0, printf 283    CAL 0, 0 284 L16: 285    LOD 1, 12 286    LDA 1, 12 287    LDX 0, 0 288    INCI 0, 0 289    STO 0, 0 290    POP 0, 1 291    JMP 0, L17 292 L18: 293    INT 0, 12 294    LDA 0, 76 295    POP 0, 4 296    ADDR 0, printf 297    CAL 0, 0 298    RET 0, 0 299 .literal 52 "%d " 300 .literal 60 "\n" 301 .literal 68 "%d " 302 .literal 76 "\n" </pre>
--	--	--

(생략)

- 어셈블리 프로그램을 시뮬레이터를 이용하여 실행하면 다음과 같은 결과가 나온다.

```

===== symbol =====
 1: printf -1
 2: malloc -2
 3: scanf -3
 4: main 128
 5: qsort 3
 6: L3 25
 7: L5 25
 8: L6 41
 9: L4 40
10: L8 41
11: L9 57
12: L7 56
13: L10 103
14: L2 103
15: L1 107
16: L11 117
17: L12 127
18: L14 213
19: L15 235
20: L13 228
21: L17 250
22: L18 272
23: L16 265
===== code =====
 0: INT 0,84
 1: SUP 0,128
 2: RET 0,0
 3: INT 0,40
 4: LDA 1,20
 5: LOD 1,12
 6: STX 0,0
 7: POP 0,1
 8: LDA 1,24

```

(생략)

```

257: LOD 1,12
258: LITI 0,4
259: MULI 0,0
260: OFFSET 0,0
261: LDI 0,0
262: POP 0,5
263: ADDR 0,-1
264: CAL 0,0
265: LOD 1,12
266: LDA 1,12
267: LDX 0,0
268: INCI 0,0
269: STO 0,0
270: POP 0,1
271: JMP 0,250
272: INT 0,12
273: LDA 0,76
274: POP 0,4
275: ADDR 0,-1
276: CAL 0,0
277: RET 0,0
start execution
0 1 3 5 7 9 2 4 6 8
0 1 2 3 4 5 6 7 8 9
end execution
minjeong@minjeong-VirtualBox

```

test16.c를 gcc로 실행하면 다음과 같은 결과가 나온다. 두 개가 동일한 결과라는 것을 확인할 수 있다.

```

minjeong@minjeong-VirtualBox:~/mjeong/hw_final$ gcc -o gcc.exe test16.c
test16.c: In function 'main':
test16.c:30:20: warning: implicit declaration of function 'printf' [-Wimplicit-function-declaration]
    for(k=0;k<10;k++) printf("%d ",a[k]);printf("\n");
                      ^
test16.c:30:20: warning: incompatible implicit declaration of built-in function 'printf'
test16.c:30:20: note: include '<stdio.h>' or provide a declaration of 'printf'
test16.c:30:39: warning: incompatible implicit declaration of built-in function 'printf'
    for(k=0;k<10;k++) printf("%d ",a[k]);printf("\n");
                              ^
test16.c:30:39: note: include '<stdio.h>' or provide a declaration of 'printf'
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$ ./gcc.exe
0 1 3 5 7 9 2 4 6 8
0 1 2 3 4 5 6 7 8 9
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$

```

### 3. 작성한 코드

```
-code_generator.c
```

```
#include <stdio.h>
```

```
#include <string.h>
```

```
#include "type.h"
```

```
typedef enum op {OP_NULL, LOD, LDX, LDXB, LDA, LITI, STO, STOB, STX, STXB,  
SUBI, SUBF, DIVI, DIVF, ADDI, ADDF, OFFSET, MULI, MULF, MOD, LSSI, LSSF,  
GTRI, GTRF, LEQI, LEQF, GEQI, GEQF, NEQI, NEQF, EQLI, EQLF, NOT, OR, AND,  
CVTI, CVTF, JPC, JPCR, JMP, JPT,JPTR, INT, INCI, INCF, DECI, DECF, SUP, CAL,  
ADDR, RET, MINUSI, MINUSF, CHK, LDI, LDIB, POP, POPB} OPCODE;
```

```
char *opcode_name[] = {"OP_NULL", "LOD", "LDX", "LDXB", "LDA", "LITI", "STO",  
"STOB", "STX", "STXB", "SUBI", "SUBF", "DIVI", "DIVF", "ADDI", "ADDF", "OFFSET",  
"MULI", "MULF", "MOD", "LSSI", "LSSF", "GTRI", "GTRF", "LEQI", "LEQF", "GEQI", "GEQF",  
"NEQI", "NEQF", "EQLI", "EQLF", "NOT", "OR", "AND", "CVTI", "CVTF", "JPC", "JPCR",  
"JMP", "JPT","JPTR", "INT", "INCI", "INCF", "DECI", "DECF", "SUP", "CAL", "ADDR", "RET",  
"MINUSI", "MINUSF", "CHK", "LDI", "LDIB", "POP", "POPB"};
```

```
void code_generation(A_NODE *);
```

```
void gen_literal_table();
```

```
void gen_program (A_NODE *);
```

```
void gen_expression (A_NODE *);
```

```
void gen_expression_left (A_NODE *);
```

```
int gen_arg_expression (A_NODE *);
```

```
void gen_statement (A_NODE *,int, int);
```

```
void gen_statement_list (A_NODE *,int, int);
```

```
void gen_declaration_list (A_ID *);
```

```
void gen_declaration (A_ID *);
```

```
void gen_code_i(OPCODE,int,int);

void gen_code_f(OPCODE,int,float);

void gen_code_s(OPCODE,int,char*);

void gen_code_l(OPCODE,int,int);

void gen_label_number(int);

void gen_label_name(char*);

void gen_error();

int get_label();

int label_no = 0;

int gen_err = 0;

extern FILE *fout;

extern A_TYPE *int_type, *char_type, *void_type, *float_type, *string_type;

extern A_LITERAL literal_table[];

extern int literal_no;


void code_generation(A_NODE *node) {

    gen_program(node);

    gen_literal_table();

}
```



```

void gen_literal_table() {

    int i;

    for (i=1;i<=literal_no; i++) {

        fprintf(fout,".literal %5d  ",literal_table[i].addr);

        if(literal_table[i].type == int_type)

            fprintf(fout,"%d\n",literal_table[i].value.i);

        else if(literal_table[i].type == float_type)

            fprintf(fout,"%f\n",literal_table[i].value.f);

        else if(literal_table[i].type == char_type)

            fprintf(fout,"%d\n",literal_table[i].value.c);

        else if(literal_table[i].type == string_type)

            fprintf(fout,"%s\n",literal_table[i].value.s);

    }

}

```

```

void gen_program(A_NODE *node) {

    switch(node->name) {

        case N_PROGRAM :

            gen_code_i(INT,0,node->value);

            gen_code_s(SUP,0,"main");

            gen_code_i(RET,0,0);

    }

}

```

```

        gen_declaration_list(node->clink);

        break;

    default:

        gen_error(100,node->line);

        break;

    }

}

```

```

void gen_expression (A_NODE *node){

    A_ID *id;

    A_TYPE *t;

    int i, ll;

    switch (node->name) {

    case N_EXP_IDENT:

        id = node->clink;

        t = id->type;

        switch (id->kind) {

        case ID_VAR:

        case ID_PARM:

            switch (t->kind)

```

```

{

case T_ENUM:

case T_POINTER:

    gen_code_i(LOD, id->level, id->address);

    break;

case T_ARRAY:

    if (id->kind == ID_VAR)

        gen_code_i(LDA, id->level, id->address);

    else

        gen_code_i(LOD, id->level, id->address);

    break;

case T_STRUCT:

case T_UNION:

    break;

default:

    gen_error(11, id->line);

    break;

}

break;

case ID_ENUM_LITERAL:

    gen_code_i(LITI, 0, id->init);

```

```

        break;

    default:

        gen_error(11, node->line);

        break;

    }

    break;

case N_EXP_INT_CONST:

    gen_code_i(LITI, 0, node->clink);

    break;

case N_EXP_FLOAT_CONST:

    i = node->clink;

    gen_code_i(LOD, 0, literal_table[i].addr);

    break;

case N_EXP_CHAR_CONST:

    gen_code_i(LITI, 0, node->clink);

    break;

case N_EXP_STRING_LITERAL:

    i = node->clink;

    gen_code_i(LDA, 0, literal_table[i].addr);

    break;

```



```

case N_EXP_ARRAY:

    gen_expression(node->llink);

    gen_expression(node->rlink);

    if (node->type->size > 1) {

        gen_code_i(LITI, 0, node->type->size);

        gen_code_i(MULI, 0, 0);

    }

    gen_code_i(OFFSET, 0, 0);

    if (!isArrayType(node->type)) {

        i = node->type->size;

        if (i == 1)

            gen_code_i(LDIB, 0, 0);

        else

            gen_code_i(LDI, 0, 0);

    }

    break;

case N_EXP_FUNCTION_CALL:

    t = node->llink->type;

    i = t->element_type->element_type->size;

    if (i % 4) i = i / 4 * 4 + 4;

    if (node->rlink) {

```

```

        gen_code_i(INT, 0, 12 + i);

        gen_arg_expression(node->rlink);

        gen_code_i(POP, 0, node->rlink->value / 4 + 3);

    }

    else

        gen_code_i(INT, 0, i);

        gen_expression(node->llink);

        gen_code_i(CAL, 0, 0);

        break;

case N_EXP_STRUCT:

        break;

case N_EXP_ARROW:

        break;

case N_EXP_POST_INC:

        gen_expression(node->clink);

        gen_expression_left(node->clink);

        t = node->type;

        if (node->type->size == 1)

            gen_code_i(LDXB, 0, 0);

        else

```

```

        gen_code_i(LDX, 0, 0);

    if (isPointerOrArrayType(node->type)) {

        gen_code_i(LITI, 0, node->type->element_type->size);

        gen_code_i(ADDI, 0, 0);

    }

    else if (isFloatType(node->type))

        gen_code_i(INCF, 0, 0);

    else

        gen_code_i(INCI, 0, 0);

    if (node->type->size == 1)

        gen_code_i(STOB, 0, 0);

    else

        gen_code_i(STO, 0, 0);

    break:

case N_EXP_POST_DEC:

    gen_expression(node->clink);

    gen_expression_left(node->clink);

    t = node->type;

    if (node->type->size == 1)

        gen_code_i(LDXB, 0, 0);

    else

```

```

        gen_code_i(LDX, 0, 0);

    if (isPointerOrArrayType(node->type)) {

        gen_code_i(LITI, 0, node->type->element_type->size);

        gen_code_i(SUBI, 0, 0);

    }

    else if (isFloatType(node->type))

        gen_code_i(DECF, 0, 0);

    else

        gen_code_i(DECI, 0, 0);

    if (node->type->size == 1)

        gen_code_i(STOB, 0, 0);

    else

        gen_code_i(STO, 0, 0);

    break;

case N_EXP_PRE_INC:

    gen_expression_left(node->clink);

    t = node->type;

    if (node->type->size == 1)

        gen_code_i(LDXB, 0, 0);

    else

```

```

        gen_code_i(LDX, 0, 0);

    if (isPointerOrArrayType(node->type)) {

        gen_code_i(LITI, 0, node->type->element_type->size);

        gen_code_i(ADDI, 0, 0);

    }

    else if (isFloatType(node->type))

        gen_code_i(INCF, 0, 0);

    else

        gen_code_i(INCI, 0, 0);

    if (node->type->size == 1)

        gen_code_i(STXB, 0, 0);

    else

        gen_code_i(STX, 0, 0);

    break:

case N_EXP_PRE_DEC:

    gen_expression_left(node->clink);

    t = node->type;

    if (node->type->size == 1)

        gen_code_i(LDXB, 0, 0);

    else

        gen_code_i(LDX, 0, 0);

```



```

if (isPointerOrArrayType(node->type)) {

    gen_code_i(LITI, 0, node->type->element_type->size);

    gen_code_i(SUBI, 0, 0);

}

else if (isFloatType(node->type))

    gen_code_i(DECF, 0, 0);

else

    gen_code_i(DECI, 0, 0);

if (node->type->size == 1)

    gen_code_i(STXB, 0, 0);

else

    gen_code_i(STX, 0, 0);

break;

case N_EXP_NOT:

    gen_expression(node->clink);

    gen_code_i(NOT, 0, 0);

    break;

case N_EXP_PLUS:

    gen_expression(node->clink);

    break;

```

```
case N_EXP_MINUS:

    gen_expression(node->clink);

    if (isFloatType(node->type))

        gen_code_i(MINUSF, 0, 0);

    else

        gen_code_i(MINUSI, 0, 0);

    break;

case N_EXP_AMP:

    gen_expression_left(node->clink);

    break;

case N_EXP_STAR:

    gen_expression(node->clink);

    i = node->type->size;

    if (i == 1)

        gen_code_i(LDIB, 0, 0);

    else

        gen_code_i(LDI, 0, 0);

    break;

case N_EXP_SIZE_TYPE:

    gen_code_i(LITI, 0, node->clink);

    break;
```

```

case N_EXP_SIZE_EXP:

    gen_code_i(LITI, 0, node->clink);

    break;

case N_EXP_CAST:

    gen_expression(node->rlink);

    if (node->type != node->rlink->type)

        if (isFloatType(node->type))

            gen_code_i(CVTF, 0, 0);

        else if (isFloatType(node->rlink->type))

            gen_code_i(CVTI, 0, 0);

    break;

case N_EXP_MUL:

    gen_expression(node->llink);

    gen_expression(node->rlink);

    if (isFloatType(node->type))

        gen_code_i(MULF, 0, 0);

    else

        gen_code_i(MULI, 0, 0);

    break;

case N_EXP_DIV:

```

```

        gen_expression(node->llink);

        gen_expression(node->rlink);

        if (isFloatType(node->type))

            gen_code_i(DIVF, 0, 0);

        else

            gen_code_i(DIVI, 0, 0);

        break;

case N_EXP_MOD:

        gen_expression(node->llink);

        gen_expression(node->rlink);

        gen_code_i(MOD, 0, 0);

        break;

case N_EXP_ADD:

        gen_expression(node->llink);

        if (isPointerOrArrayType(node->rlink->type)) {

            gen_code_i(LITI, 0, node->rlink->type->element_type->size);

            gen_code_i(MULI, 0, 0);

        }

        gen_expression(node->rlink);

        if (isPointerOrArrayType(node->llink->type)) {

            gen_code_i(LITI, 0, node->llink->type->element_type->size);

```

```

        gen_code_i(MULI, 0, 0);

    }

    if (isFloatType(node->type))

        gen_code_i(ADDF, 0, 0);

    else

        gen_code_i(ADDI, 0, 0);

    break;

case N_EXP_SUB:

    gen_expression(node->llink);

    gen_expression(node->rlink);

    if      (isPointerOrArrayType(node->llink->type)      &&
!isPointerOrArrayType(node->rlink->type)) {

        gen_code_i(LITI, 0, node->llink->type->element_type->size);

        gen_code_i(MULI, 0, 0);

    }

    if (isFloatType(node->type))

        gen_code_i(SUBF, 0, 0);

    else

        gen_code_i(SUBI, 0, 0);

    break;

case N_EXP_LSS:

```



```

        gen_expression(node->llink);

        gen_expression(node->rlink);

        if (isFloatType(node->llink->type))

            gen_code_i(LSSF, 0, 0);

        else

            gen_code_i(LSSI, 0, 0);

        break;

case N_EXP_GTR:

        gen_expression(node->llink);

        gen_expression(node->rlink);

        if (isFloatType(node->llink->type))

            gen_code_i(GTRF, 0, 0);

        else

            gen_code_i(GTRI, 0, 0);

        break;

case N_EXP_LEQ:

        gen_expression(node->llink);

        gen_expression(node->rlink);

        if (isFloatType(node->llink->type))

            gen_code_i(LEQF, 0, 0);

```

```

        else

            gen_code_i(LEQI, 0, 0);

        break;

case N_EXP_GEQ:

    gen_expression(node->llink);

    gen_expression(node->rlink);

    if (isFloatType(node->llink->type))

        gen_code_i(GEQF, 0, 0);

    else

        gen_code_i(GEQI, 0, 0);

    break;

case N_EXP_NEQ:

    gen_expression(node->llink);

    gen_expression(node->rlink);

    if (isFloatType(node->llink->type))

        gen_code_i(NEQF, 0, 0);

    else

        gen_code_i(NEQI, 0, 0);

    break;

case N_EXP_EQL:

    gen_expression(node->llink);

```

```

        gen_expression(node->rlink);

        if (isFloatType(node->llink->type))

            gen_code_i(EQLF, 0, 0);

        else

            gen_code_i(EQLI, 0, 0);

        break;

case N_EXP_AND:

    gen_expression(node->llink);

    gen_code_l(JPCR, 0, i = get_label());

    gen_expression(node->rlink);

    gen_label_number(i);

    break;

case N_EXP_OR:

    gen_expression(node->llink);

    gen_code_l(JPTR, 0, i = get_label());

    gen_expression(node->rlink);

    gen_label_number(i);

    break;

case N_EXP_ASSIGN:

    gen_expression_left(node->llink);

```

```

        gen_expression(node->rlink);

        i = node->type->size;

        if (i == 1)

            gen_code_i(STXB, 0, 0);

        else

            gen_code_i(STX, 0, 0);

        break;

    default:

        gen_error(100, node->line);

        break;

}

}

```

```

void gen_expression_left (A_NODE *node){

```

```

    A_ID *id;

```

```

    A_TYPE *t;

```

```

    int result;

```

```

    switch (node->name) {

```

```

    case N_EXP_IDENT:

```

```

        id = node->clink;

```

```

        t = id->type;

```

```

        switch (id->kind) {

```



```

case ID_VAR:

case ID_PARM:

    switch (t->kind){

    case T_ENUM:

    case T_POINTER:

        gen_code_i(LDA, id->level, id->address);

        break;

    case T_ARRAY:

        if (id->kind == ID_VAR)

            gen_code_i(LDA, id->level, id->address);

        else

            gen_code_i(LOD, id->level, id->address);

        break;

    case T_STRUCT:

    case T_UNION:

        break;

    default:

        gen_error(13, node->line,id->name);

        break;

    }

```

```

        break;

case ID_FUNC:

    gen_code_s(ADDR, 0, id->name);

    break;

default:

    gen_error(13, node->line, id->name);

    break;

}

break;

case N_EXP_ARRAY:

    gen_expression(node->llink);

    gen_expression(node->rlink);

    if (node->type->size > 1) {

        gen_code_i(LITI, 0, node->type->size);

        gen_code_i(MULI, 0, 0);

    }

    gen_code_i(OFFSET, 0, 0);

    break;

case N_EXP_STRUCT:

    break;

case N_EXP_ARROW:

```

```
        break;

case N_EXP_STAR:

    gen_expression(node->clink);

    break;

case N_EXP_INT_CONST:

case N_EXP_FLOAT_CONST:

case N_EXP_CHAR_CONST:

case N_EXP_STRING_LITERAL:

case N_EXP_FUNCTION_CALL:

case N_EXP_POST_INC:

case N_EXP_POST_DEC:

case N_EXP_PRE_INC:

case N_EXP_PRE_DEC:

case N_EXP_NOT:

case N_EXP_MINUS:

case N_EXP_CAST:

case N_EXP_SIZE_TYPE:

case N_EXP_SIZE_EXP:

case N_EXP_MUL:

case N_EXP_DIV:
```

```

    case N_EXP_MOD:

    case N_EXP_ADD:

    case N_EXP_SUB:

    case N_EXP_LSS:

    case N_EXP_GTR:

    case N_EXP_LEQ:

    case N_EXP_GEQ:

    case N_EXP_NEQ:

    case N_EXP_EQL:

    case N_EXP_AMP:

    case N_EXP_AND:

    case N_EXP_OR:

    case N_EXP_ASSIGN:

        gen_error(12, node->line);

        break;

    default:

        gen_error(100, node->line);

        break;

}

}

int gen_arg_expression (A_NODE *node){

```



```

A_NODE *n;

switch(node->name){

    case N_ARG_LIST:

        gen_expression(node->llink);

        gen_arg_expression(node->rlink);

        break;

    case N_ARG_LIST_NIL:

        break;

    default:

        gen_error(100,node->line);

        break;

}

}

int get_label(){

    label_no++;

    return(label_no);

}

void gen_statement (A_NODE *node, int cnt_label, int brk_label)

```

```
{
```

```
    A_NODE *n;
```

```
    int i,l1,l2,l3;
```

```
    switch(node->name) {
```

```
        case N_STMT_LABEL_CASE :
```

```
        case N_STMT_LABEL_DEFAULT :
```

```
            break;
```

```
        case N_STMT_COMPOUND:
```

```
            if(node->llink) gen_declaration_list(node->llink);
```

```
            gen_statement_list(node->rlink ,cnt_label, brk_label);
```

```
            break;
```

```
        case N_STMT_EMPTY:
```

```
            break;
```

```
        case N_STMT_EXPRESSION:
```

```
            n=node->clink;
```

```
            gen_expression(n);
```

```
            i=n->type->size;
```

```
            if (i) gen_code_i(POP,0,i%4?i/4+1:i/4);
```

```
            break;
```

```
        case N_STMT_IF:
```

```
            gen_expression(node->llink);
```

```

        gen_code_l(JPC, 0, l1=get_label());

        gen_statement(node->rlink,cnt_label,brk_label);

        gen_label_number(l1);

        break;

case N_STMT_IF_ELSE:

        gen_expression(node->llink);

        gen_code_l(JPC, 0, l1=get_label());

        gen_statement(node->clink,cnt_label,brk_label);

        gen_code_l(JMP, 0, l2=get_label());

        gen_label_number(l1);

        gen_statement(node->rlink,cnt_label,brk_label);

        gen_label_number(l2);

        break;

case N_STMT_SWITCH:

        break;

case N_STMT_WHILE:

        l3=get_label();

        gen_label_number(l1=get_label());

        gen_expression(node->llink);

        gen_code_l(JPC, 0, l2=get_label());

```

```
gen_statement(node->rlink,l3,l2);
```

```
gen_label_number(l3);
```

```
gen_code_l(JMP, 0, 11);
```

```
gen_label_number(l2);
```

```
break;
```

```
case N_STMT_DO:
```

```
l3=get_label();
```

```
l2=get_label();
```

```
gen_label_number(l1=get_label());
```

```
gen_statement(node->llink,l2,l3);
```

```
gen_label_number(l2);
```

```
gen_expression(node->rlink);
```

```
gen_code_l(JPT, 0, 11);
```

```
gen_label_number(l3);
```

```
break;
```

```
case N_STMT_FOR:
```

```
n=node->llink;
```

```
l3=get_label();
```

```
if (n->llink){
```

```
    gen_expression(n->llink);
```

```
    i=n->llink->type->size;
```

```

        if(i)

            gen_code_i(POP,0,i%4?i/4+1:i/4);

    }

    gen_label_number(l1=get_label());

    l2=get_label();

    if (n->clink) {

        gen_expression(n->clink);

        gen_code_l(JPC, 0, l2);

    }

    gen_statement(node->rlink,l3,l2);

    gen_label_number(l3);

    if (n->rlink){

        gen_expression(n->rlink);

        i=n->rlink->type->size;

        if(i)

            gen_code_i(POP,0,i%4?i/4+1:i/4);

    }

    gen_code_l(JMP, 0, l1);

    gen_label_number(l2);

    break;

```

```

case N_STMT_CONTINUE:

    if (cnt_label)

        gen_code_l(JMP,0,cnt_label);

    else

        gen_error(22,node->line);

    break;

case N_STMT_BREAK:

    if (brk_label)

        gen_code_l(JMP,0,brk_label);

    else

        gen_error(23,node->line);

    break;

case N_STMT_RETURN:

    n=node->clink;

    if(n) { i=n->type->size;

        if (i%4) i=i/4*4+4;

        gen_code_i(LDA, 1, -i);

        gen_expression(n);

        gen_code_i(STO, 0, 0);}

    //gen_code_i(RET,0,0);

    break;

```



```

        default:

            gen_error(100,node->line);

            break;

    }

}

void gen_statement_list (A_NODE *node,int cnt_label, int brk_label){

    switch (node->name) {

        case N_STMT_LIST:

            gen_statement(node->llink, cnt_label, brk_label);

            gen_statement_list(node->rlink, cnt_label, brk_label);

            break;

        case N_STMT_LIST_NIL:

            break;

        default:

            gen_error(100, node->line);

            break;

    }

}

```

```

void gen_declaration_list (A_ID *id){

    while(id){

        gen_declaration(id);

        id=id->link;

    }

}

```

```

void gen_declaration(A_ID *id) {

    int i;

    A_NODE *node;

    switch (id->kind) {

        case ID_VAR:

            break;

        case ID_FUNC:

            if (id->type->expr) {

                gen_label_name(id->name);

                gen_code_i(INT, 0, id->type->local_var_size);

                gen_statement(id->type->expr, 0, 0);

                gen_code_i(RET,0,0); }

            break;

        case ID_PARM:

```

```

        case ID_TYPE:

        case ID_ENUM:

        case ID_STRUCT:

        case ID_FIELD:

        case ID_ENUM_LITERAL:

        case ID_NULL:

                break;

        default:

                gen_error(100,id->line);

                break;

    }

}

void gen_error(int i, int ll, char*s){

    gen_err++;

    printf("*** error at line %d: ",ll);

    switch(i){

        case 11: printf("illegal identifier in expression\n");

                break;

        case 12: printf("illegal l-value expression\n");

```

```

        break;

    case 13: printf("identifier %s not l-value expression\n",s);

        break;

    case 22: printf("no destination for continue statement\n");

        break;

    case 23: printf("no destination for break statement\n");

        break;

    case 100: printf("fatal compiler error during code generation\n");

        break;

    default:printf("unknown \n");

        break;
}

}

void gen_code_i(OPCODE op,int l,int a){

    fprintf(fout,"\t%9s      %d, %d\n",opcode_name[op],l,a);

}

void gen_code_f(OPCODE op,int l,float a){

    fprintf(fout,"\t%9s      %d, %f\n",opcode_name[op],l,a);

}

void gen_code_s(OPCODE op,int l,char* a){

```

```
        fprintf(fout, "\t%9s      %d, %s\n", opcode_name[op], l, a);

    }

void gen_code_l(OPCODE op, int l, int a){

    fprintf(fout, "\t%9s      %d, L%d\n", opcode_name[op], l, a);

}

void gen_label_number(int i){

    fprintf(fout, "L%d:\n", i);

}

void gen_label_name(char* s){

    fprintf(fout, "%s:\n", s);

}
```

-main.c

#include <stdio.h>

#include <stdlib.h>

#include "type.h"

extern int syntax\_err;

extern int semantic\_err;

extern A\_NODE \*root;

FILE \*fout;

extern void initialize();

extern void print\_ast();

extern void print\_sem\_ast();

extern void semantic\_analysis();

extern void code\_generation();

void main(int argc, char \*argv[])

{

if ((fout = fopen("a.asm", "w")) == NULL)

{

printf("can not open output file: a.asm\n");



```
        exit(1);

    }

    initialize();

    yyparse();

    if (syntax_err)

        exit(1);

    else

        print_ast(root);

    semantic_analysis(root);

    if(semantic_err)

        exit(1);

    else

        print_sem_ast(root);

    code_generation(root);

    exit(0);

}
```