컴파일러 최종과제

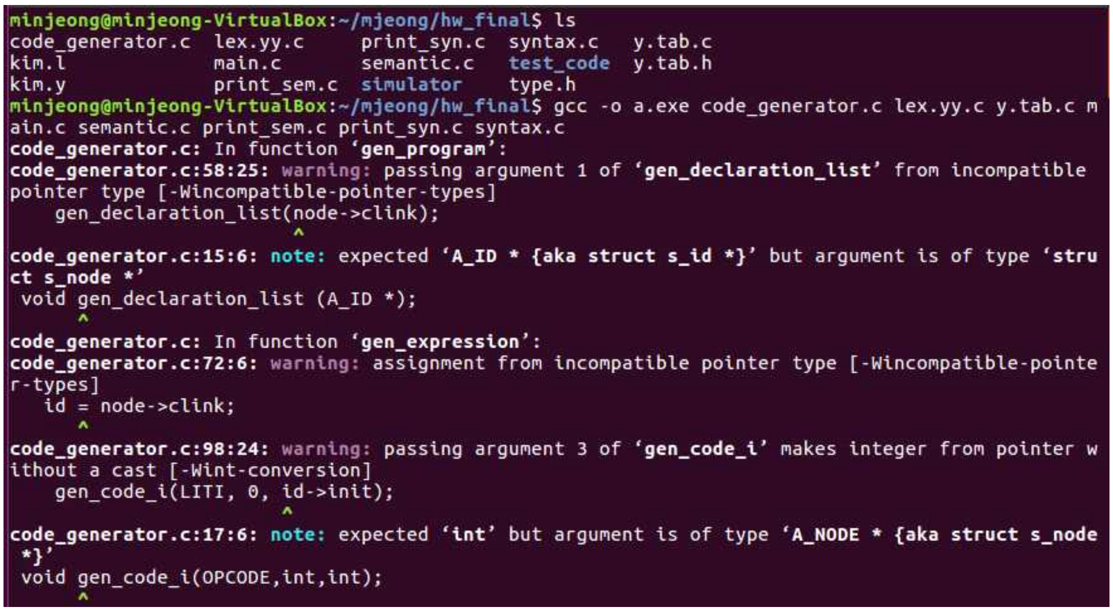
전자정보공학부
20160433
김민정

1. 실험방법

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

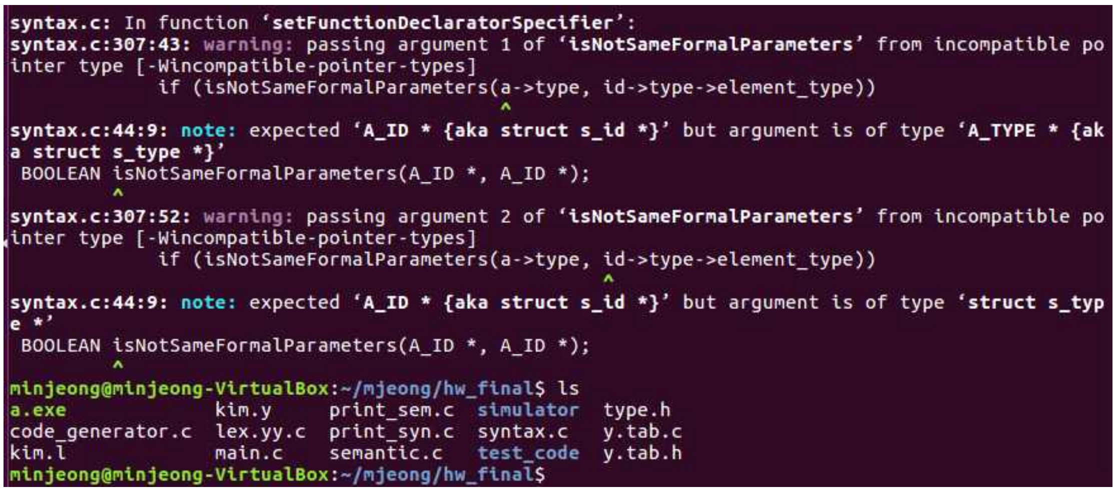
 - 원시 프로그램은 신택스 분석과 시멘틱 분석 과정을 통하여 신택스 트리와 관련 테이블들로 번역되었다. 이런 신택스 트리를 중간 언어의 형태로 번역하고 프로그램의 실행 표율을 높이기 위해 중간 언어프로그램을 최적화하는 과정을 하는 것이 바람직하나, 편의상 생략하고 신택스 트리로부터 바로 기계어 코드를 생성하고자 하였다. 시멘틱 분석과정을 통하여 모든 선언문들의 성격을 분석하고 코드 생성에 필요한 주소나 크기들이 계산되었으며 명령문들이나 수식이 올바르게 사용되고 있는지 분석하고 그 타입도 규칙에 따라 변환하여서 올바른 타입의 수식이나 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 m
ain.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 'stru
ct 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-pointe
r-types]
   id = node->clink;
      ^
code_generator.c:98:24: warning: passing argument 3 of 'gen_code_i' makes integer from pointer w
ithout 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);
      ^
```

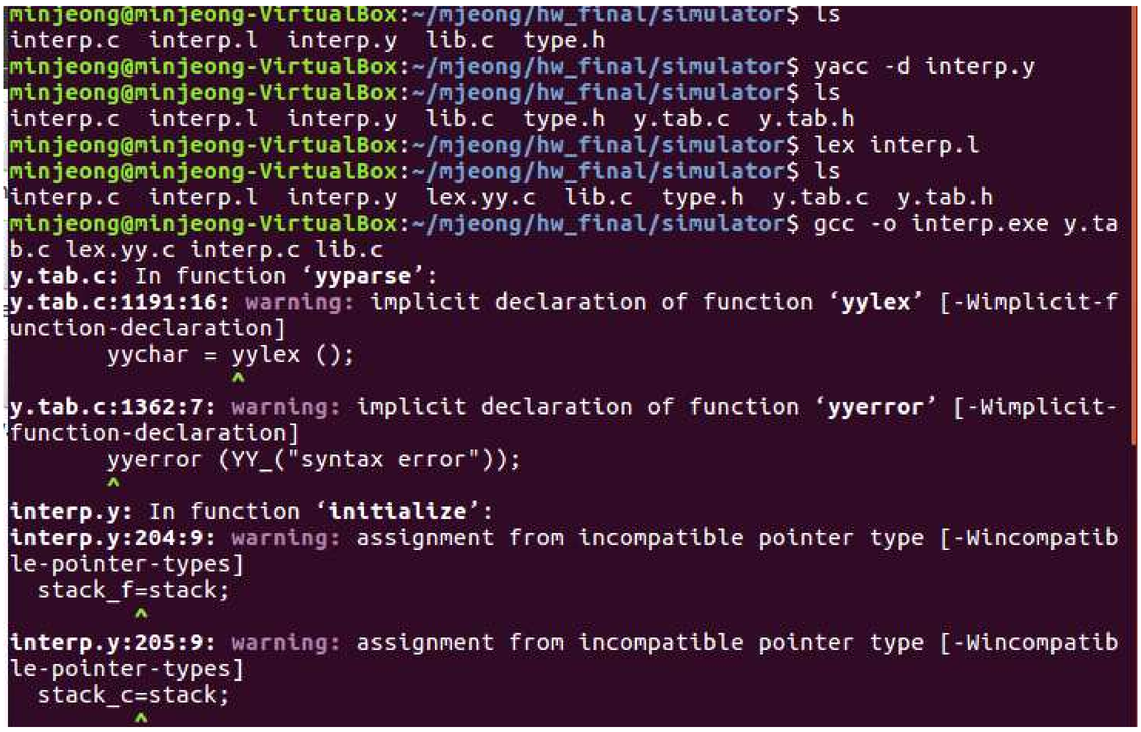
(생략)



```
syntax.c: In function 'setFunctionDeclaratorSpecifier':
syntax.c:307:43: warning: passing argument 1 of 'isNotSameFormalParameters' from incompatible po
inter 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 * {ak
a struct s_type *}'
 BOOLEAN isNotSameFormalParameters(A_ID *, A_ID *);
         ^
syntax.c:307:52: warning: passing argument 2 of 'isNotSameFormalParameters' from incompatible po
inter 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_typ
e *'
 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$
```

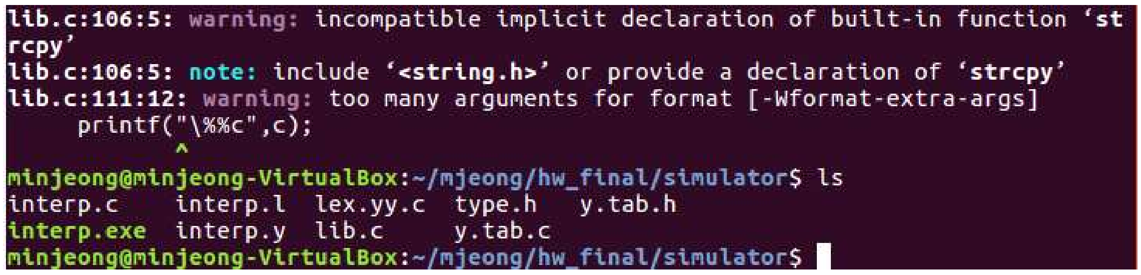
- code generator에서는 초기화구문, switch문, structure와 관련된 부분은 만들지 않았다. 따라서 예제에서는 switch나 structure가 포함되어 있지 않는 코드들을 가지고 실험을 해볼 것이다.

- 주어진 어셈블러/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.ta
b.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-f
unction-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 [-Wincompatib
le-pointer-types]
  stack_f=stack;
         ^
interp.y:205:9: warning: assignment from incompatible pointer type [-Wincompatib
le-pointer-types]
  stack_c=stack;
         ^
```

(생략)

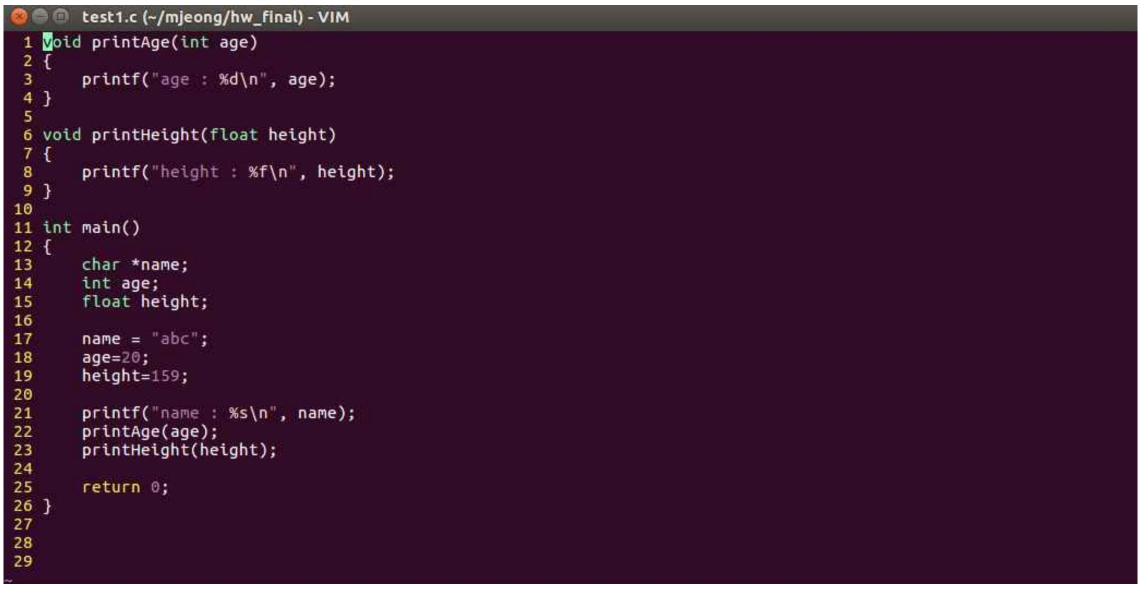


```
lib.c:106:5: warning: incompatible implicit declaration of built-in function ‘st
rcpy’
lib.c:106:5: note: include ‘<string.h>’ or provide a declaration of ‘strcpy’
lib.c:111:12: warning: too many arguments for format [-Wformat-extra-args]
     printf("\%%c",c);
            ^
minjeong@minjeong-VirtualBox:~/mjeong/hw_final/simulator$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

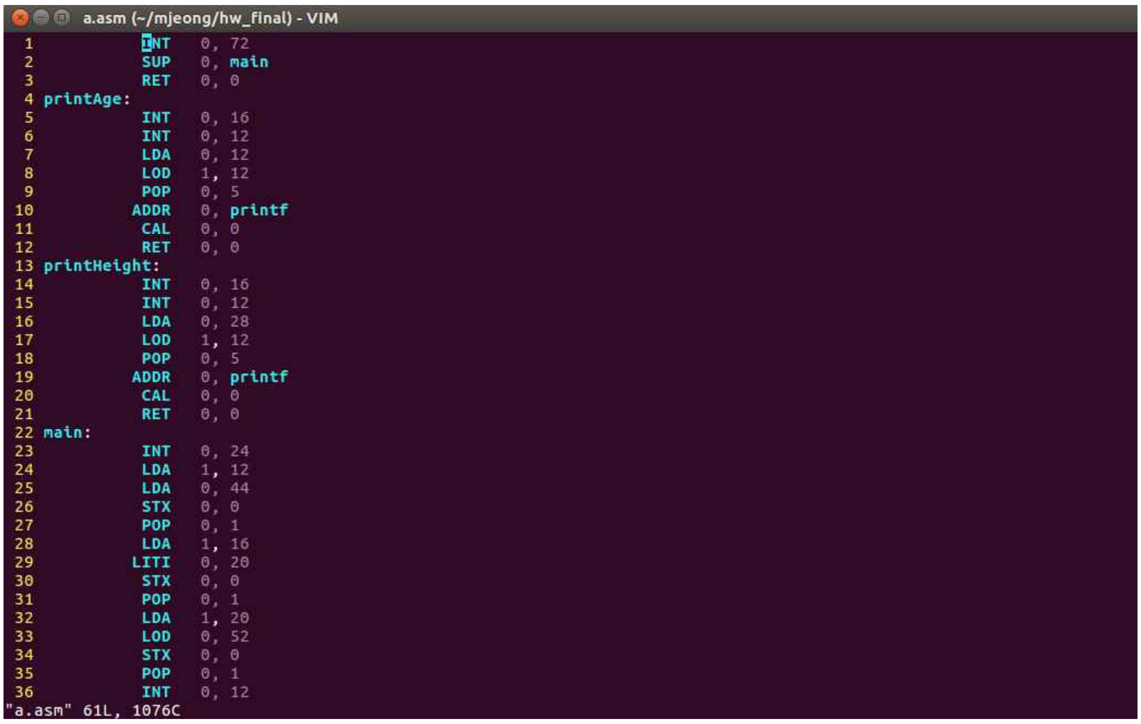
1) test1.c

이름과 나이, 키를 출력하는 프로그램이다.

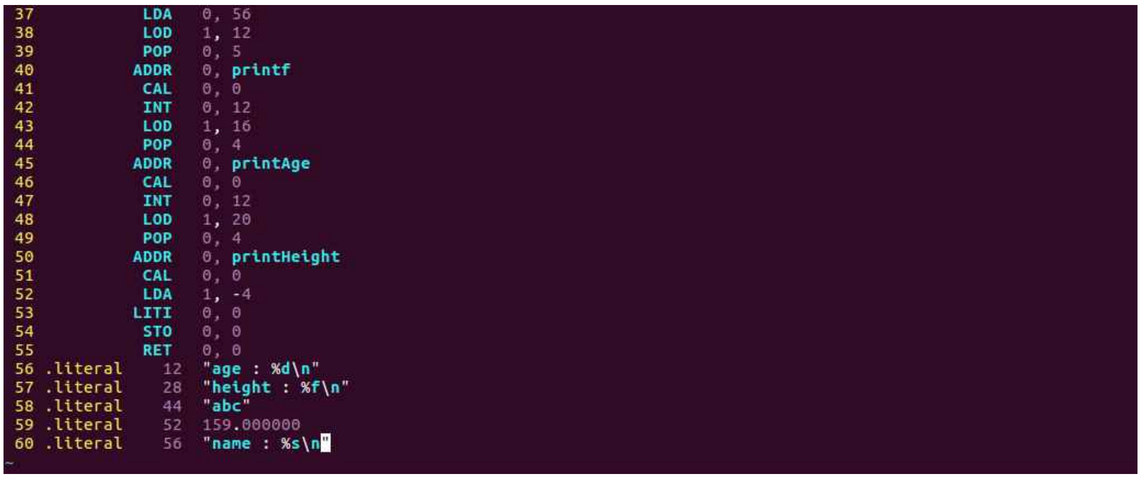


```c
void printAge(int age)
{
    printf("age : %d\n", age);
}

void printHeight(float height)
{
    printf("height : %f\n", height);
}

int main()
{
    char *name;
    int age;
    float height;

    name = "abc";
    age=20;
    height=159;

    printf("name : %s\n", name);
    printAge(age);
    printHeight(height);

    return 0;
}
```

● 컴파일러를 통하여 만들어진 어셈블러 프로그램은 다음과 같다.

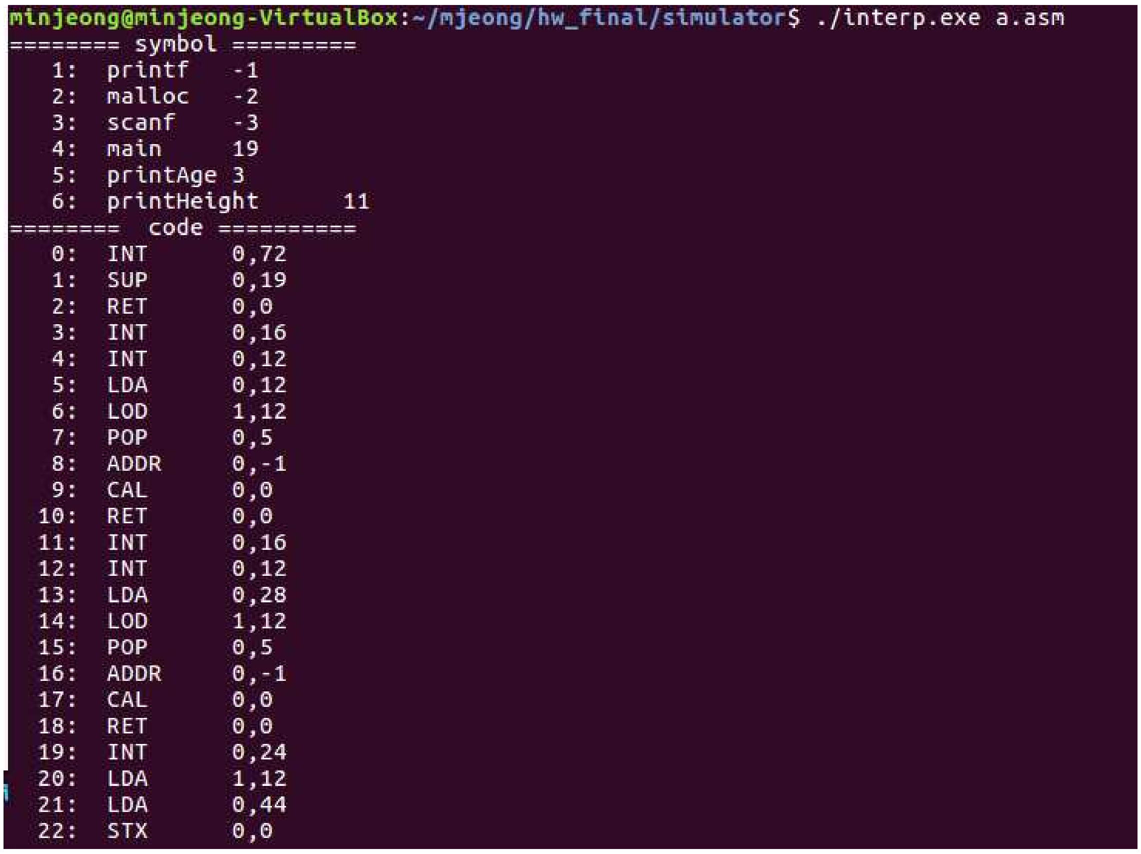


```
          INT    0, 72
          SUP    0, main
          RET    0, 0
printAge:
          INT    0, 16
          INT    0, 12
          LDA    0, 12
          LOD    1, 12
          POP    0, 5
          ADDR   0, printf
          CAL    0, 0
          RET    0, 0
printHeight:
          INT    0, 16
          INT    0, 12
          LDA    0, 28
          LOD    1, 12
          POP    0, 5
          ADDR   0, printf
          CAL    0, 0
          RET    0, 0
main:
          INT    0, 24
          LDA    1, 12
          LDA    0, 44
          STX    0, 0
          POP    0, 1
          LDA    1, 16
          LITI   0, 20
          STX    0, 0
          POP    0, 1
          LDA    1, 20
          LOD    0, 52
          STX    0, 0
          POP    0, 1
          INT    0, 12
"a.asm" 61L, 1076C
```



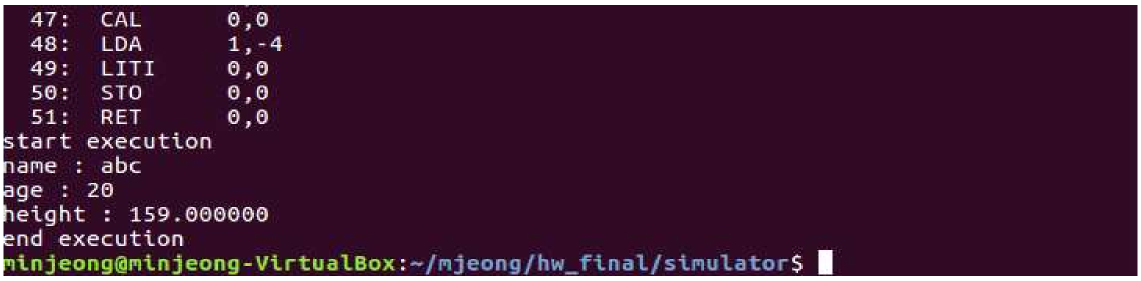
```
            LDA    0, 56
            LOD    1, 12
            POP    0, 5
           ADDR    0, printf
            CAL    0, 0
            INT    0, 12
            LOD    1, 16
            POP    0, 4
           ADDR    0, printAge
            CAL    0, 0
            INT    0, 12
            LOD    1, 20
            POP    0, 4
           ADDR    0, printHeight
            CAL    0, 0
            LDA    1, -4
           LITI    0, 0
            STO    0, 0
            RET    0, 0
.literal     12  "age : %d\n"
.literal     28  "height : %f\n"
.literal     44  "abc"
.literal     52  159.000000
.literal     56  "name : %s\n"
```

● 어셈블러 프로그램을 시뮬레이터를 이용하여 실행하면 다음과 같은 결과가 나온다.



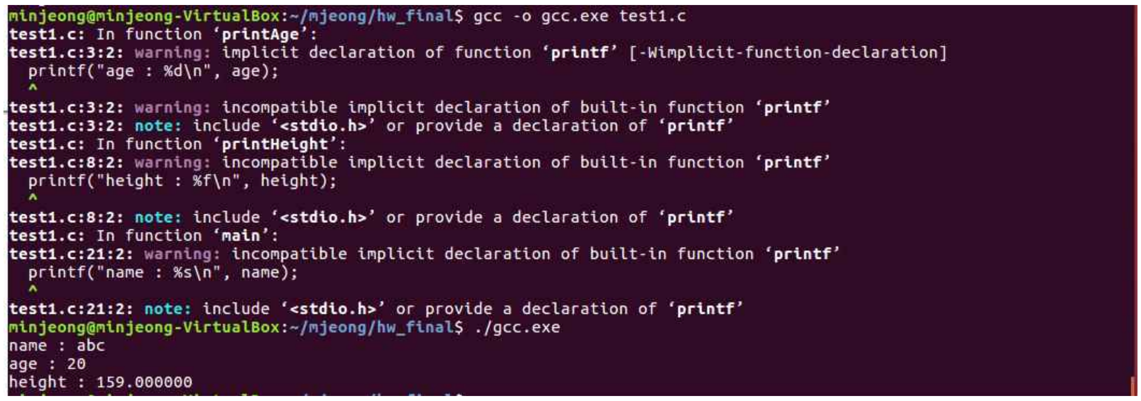
```
minjeong@minjeong-VirtualBox:~/mjeong/hw_final/simulator$ ./interp.exe a.asm
======== symbol =========
   1:   printf   -1
   2:   malloc   -2
   3:   scanf    -3
   4:   main     19
   5:   printAge 3
   6:   printHeight      11
========  code ==========
   0:   INT      0,72
   1:   SUP      0,19
   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)



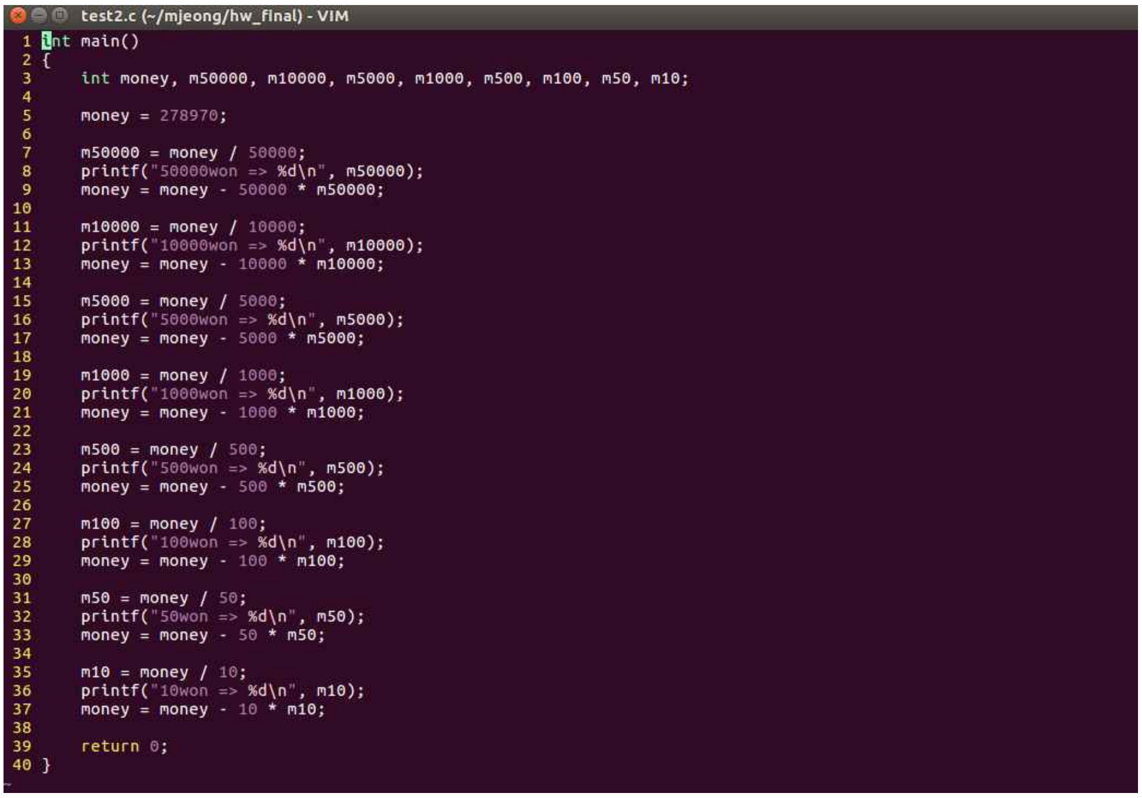
```
  47:   CAL      0,0
  48:   LDA      1,-4
  49:   LITI     0,0
  50:   STO      0,0
  51:   RET      0,0
start execution
name : abc
age : 20
height : 159.000000
end execution
minjeong@minjeong-VirtualBox:~/mjeong/hw_final/simulator$
```

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



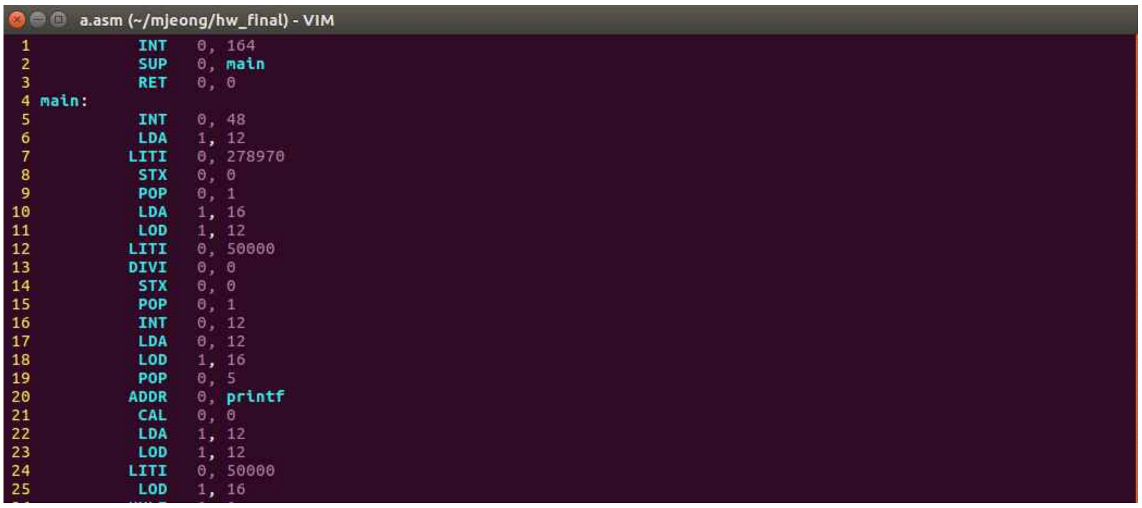
```
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$ gcc -o gcc.exe test1.c
test1.c: In function ‘printAge’:
test1.c:3:2: warning: implicit declaration of function ‘printf’ [-Wimplicit-function-declaration]
  printf("age : %d\n", age);
  ^
test1.c:3:2: warning: incompatible implicit declaration of built-in 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

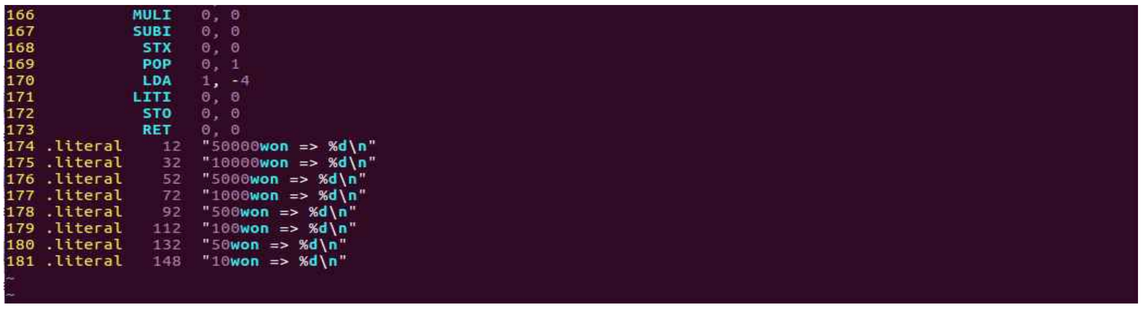
```
int main()
{
    int money, m50000, m10000, m5000, m1000, m500, m100, m50, m10;

    money = 278970;

    m50000 = money / 50000;
    printf("50000won => %d\n", m50000);
    money = money - 50000 * m50000;

    m10000 = money / 10000;
    printf("10000won => %d\n", m10000);
    money = money - 10000 * m10000;

    m5000 = money / 5000;
    printf("5000won => %d\n", m5000);
    money = money - 5000 * m5000;

    m1000 = money / 1000;
    printf("1000won => %d\n", m1000);
    money = money - 1000 * m1000;

    m500 = money / 500;
    printf("500won => %d\n", m500);
    money = money - 500 * m500;

    m100 = money / 100;
    printf("100won => %d\n", m100);
    money = money - 100 * m100;

    m50 = money / 50;
    printf("50won => %d\n", m50);
    money = money - 50 * m50;

    m10 = money / 10;
    printf("10won => %d\n", m10);
    money = money - 10 * m10;

    return 0;
}
```

● 컴파일러를 통하여 만들어진 어셈블러 프로그램은 다음과 같다.

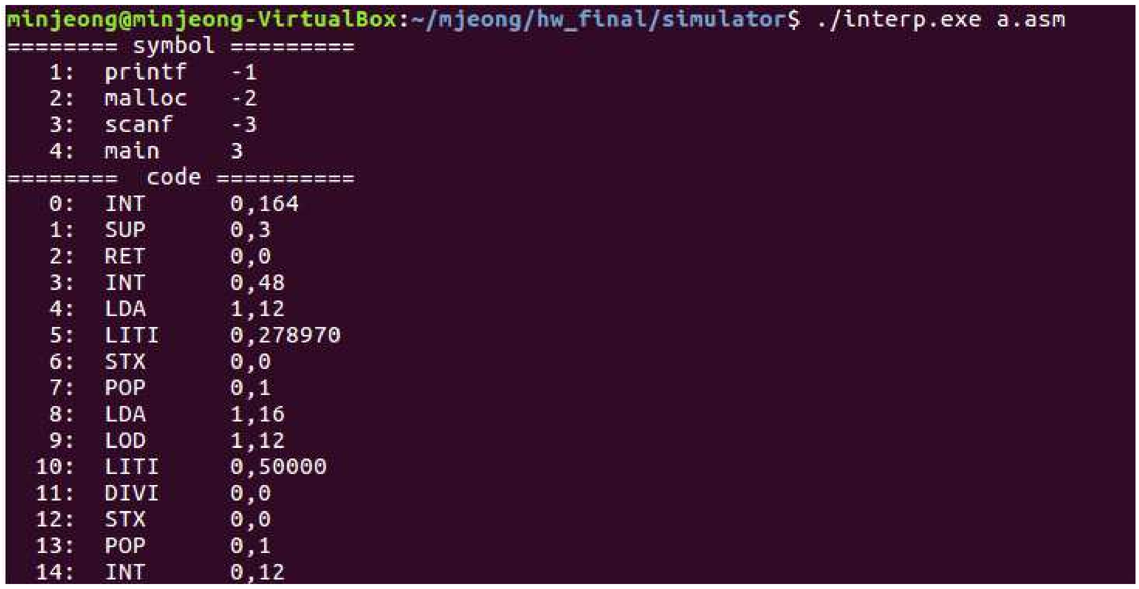

a.asm (~/mjeong/hw_final) - VIM

```
          INT    0, 164
          SUP    0, main
          RET    0, 0
main:
          INT    0, 48
          LDA    1, 12
         LITI    0, 278970
          STX    0, 0
          POP    0, 1
          LDA    1, 16
          LOD    1, 12
         LITI    0, 50000
         DIVI    0, 0
          STX    0, 0
          POP    0, 1
          INT    0, 12
          LDA    0, 12
          LOD    1, 16
          POP    0, 5
         ADDR    0, printf
          CAL    0, 0
          LDA    1, 12
          LOD    1, 12
         LITI    0, 50000
          LOD    1, 16
```

(생략)

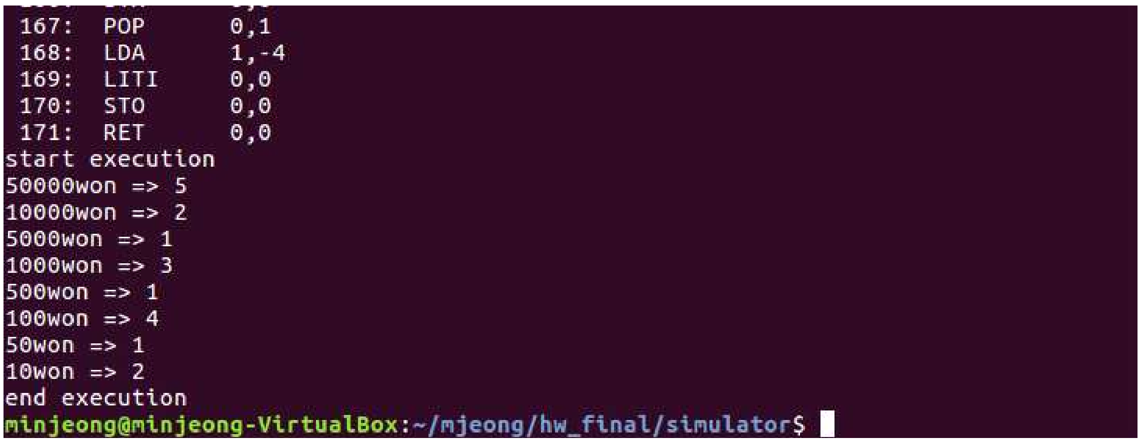

```
           MULI    0, 0
           SUBI    0, 0
            STX    0, 0
            POP    0, 1
            LDA    1, -4
           LITI    0, 0
            STO    0, 0
            RET    0, 0
 .literal     12   "50000won => %d\n"
 .literal     32   "10000won => %d\n"
 .literal     52   "5000won => %d\n"
 .literal     72   "1000won => %d\n"
 .literal     92   "500won => %d\n"
 .literal    112   "100won => %d\n"
 .literal    132   "50won => %d\n"
 .literal    148   "10won => %d\n"
```

● 어셈블러 프로그램을 시뮬레이터를 이용하여 실행하면 다음과 같은 결과가 나온다.



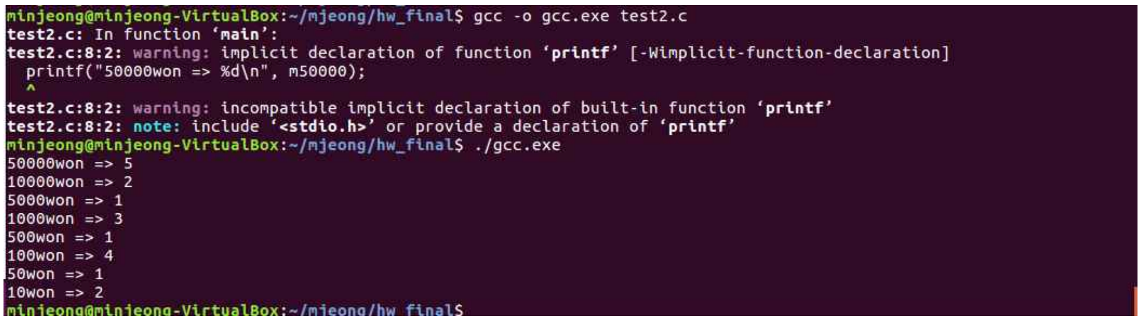
```
minjeong@minjeong-VirtualBox:~/mjeong/hw_final/simulator$ ./interp.exe a.asm
======== symbol =========
   1:  printf    -1
   2:  malloc    -2
   3:  scanf     -3
   4:  main      3
========  code ==========
   0:  INT       0,164
   1:  SUP       0,3
   2:  RET       0,0
   3:  INT       0,48
   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)



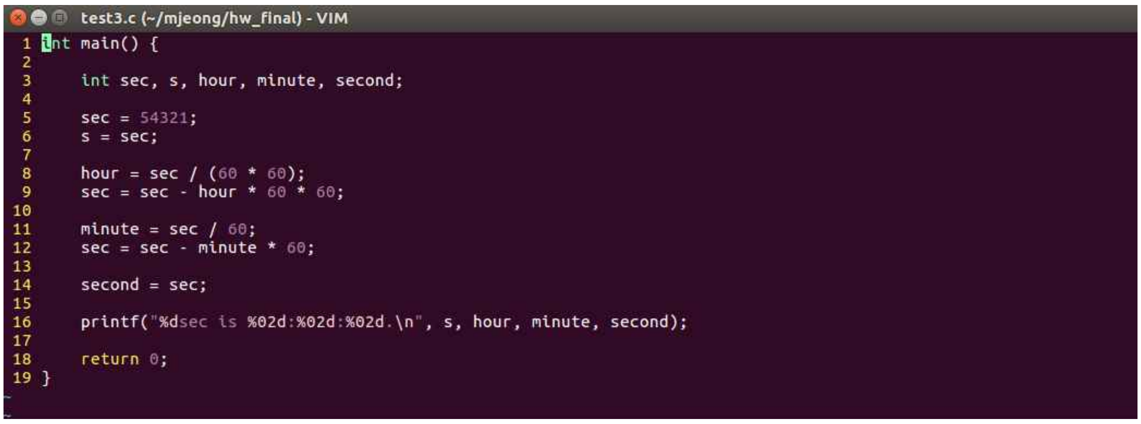
```
 167:  POP       0,1
 168:  LDA       1,-4
 169:  LITI      0,0
 170:  STO       0,0
 171:  RET       0,0
start execution
50000won => 5
10000won => 2
5000won => 1
1000won => 3
500won => 1
100won => 4
50won => 1
10won => 2
end execution
minjeong@minjeong-VirtualBox:~/mjeong/hw_final/simulator$
```

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



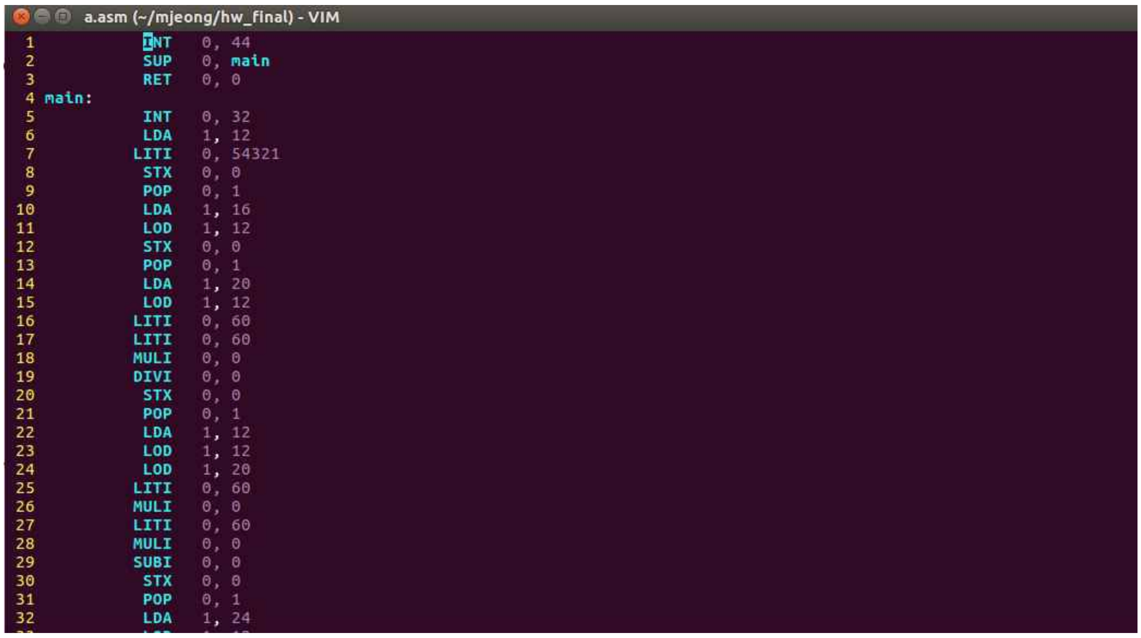
```
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$ gcc -o gcc.exe test2.c
test2.c: In function ‘main’:
test2.c:8:2: warning: implicit declaration of function ‘printf’ [-Wimplicit-function-declaration]
  printf("50000won => %d\n", m50000);
  ^
test2.c:8:2: warning: incompatible implicit declaration of 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



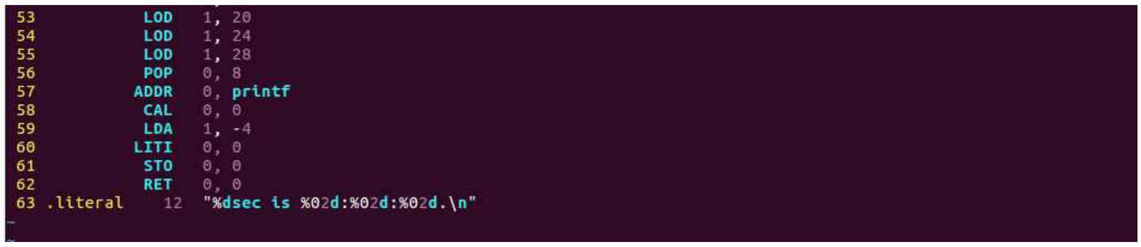
```c
int main() {

    int sec, s, hour, minute, second;

    sec = 54321;
    s = sec;

    hour = sec / (60 * 60);
    sec = sec - hour * 60 * 60;

    minute = sec / 60;
    sec = sec - minute * 60;

    second = sec;

    printf("%dsec is %02d:%02d:%02d.\n", s, hour, minute, second);

    return 0;
}
```

● 컴파일러를 통하여 만들어진 어셈블러 프로그램은 다음과 같다.



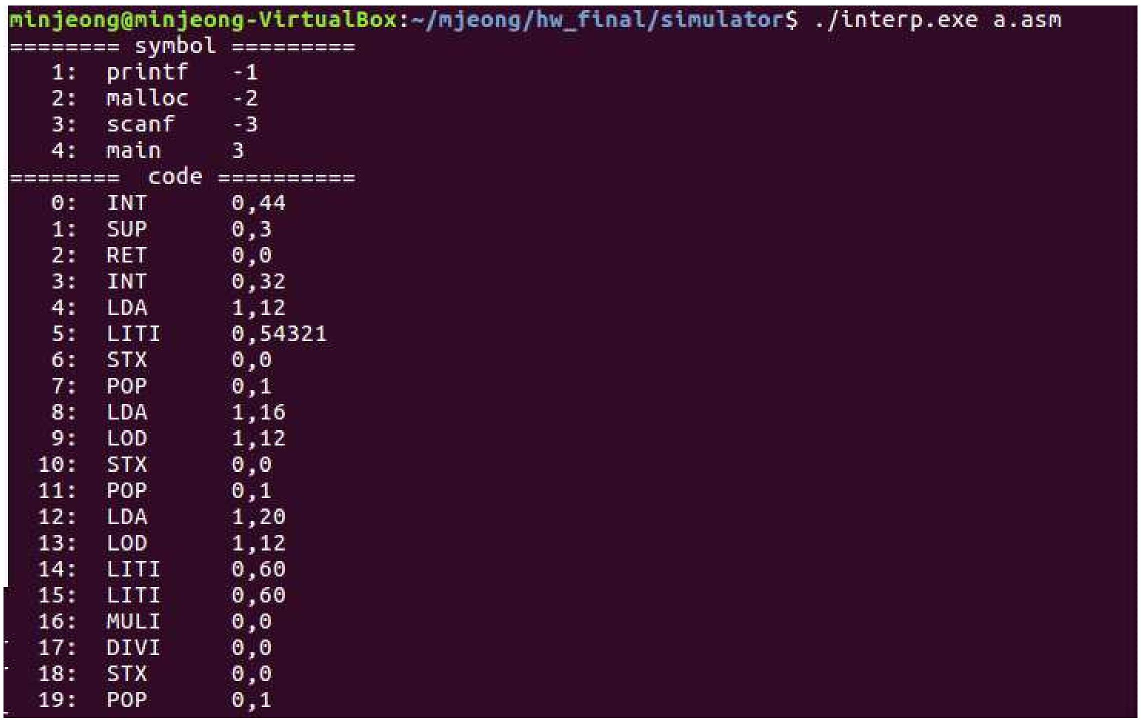
```
             INT     0, 44
             SUP     0, main
             RET     0, 0
main:
             INT     0, 32
             LDA     1, 12
            LITI     0, 54321
             STX     0, 0
             POP     0, 1
             LDA     1, 16
             LOD     1, 12
             STX     0, 0
             POP     0, 1
             LDA     1, 20
             LOD     1, 12
            LITI     0, 60
            LITI     0, 60
            MULI     0, 0
            DIVI     0, 0
             STX     0, 0
             POP     0, 1
             LDA     1, 12
             LOD     1, 12
             LOD     1, 20
            LITI     0, 60
            MULI     0, 0
            LITI     0, 60
            MULI     0, 0
            SUBI     0, 0
             STX     0, 0
             POP     0, 1
             LDA     1, 24
```

(생략)



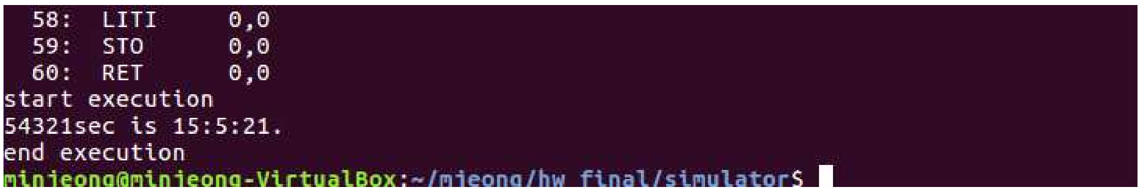
```
             LOD     1, 20
             LOD     1, 24
             LOD     1, 28
             POP     0, 8
            ADDR     0, printf
             CAL     0, 0
             LDA     1, -4
            LITI     0, 0
             STO     0, 0
             RET     0, 0
.literal       12    "%dsec is %02d:%02d:%02d.\n"
```

● 어셈블러 프로그램을 시뮬레이터를 이용하여 실행하면 다음과 같은 결과가 나온다.



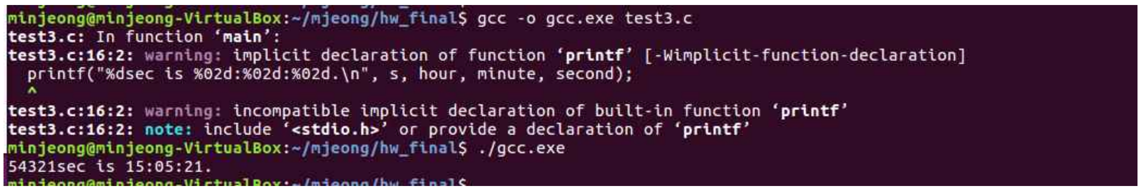
```
minjeong@minjeong-VirtualBox:~/mjeong/hw_final/simulator$ ./interp.exe a.asm
======== symbol =========
   1:  printf   -1
   2:  malloc   -2
   3:  scanf    -3
   4:  main     3
========  code ==========
   0:  INT      0,44
   1:  SUP      0,3
   2:  RET      0,0
   3:  INT      0,32
   4:  LDA      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$
```

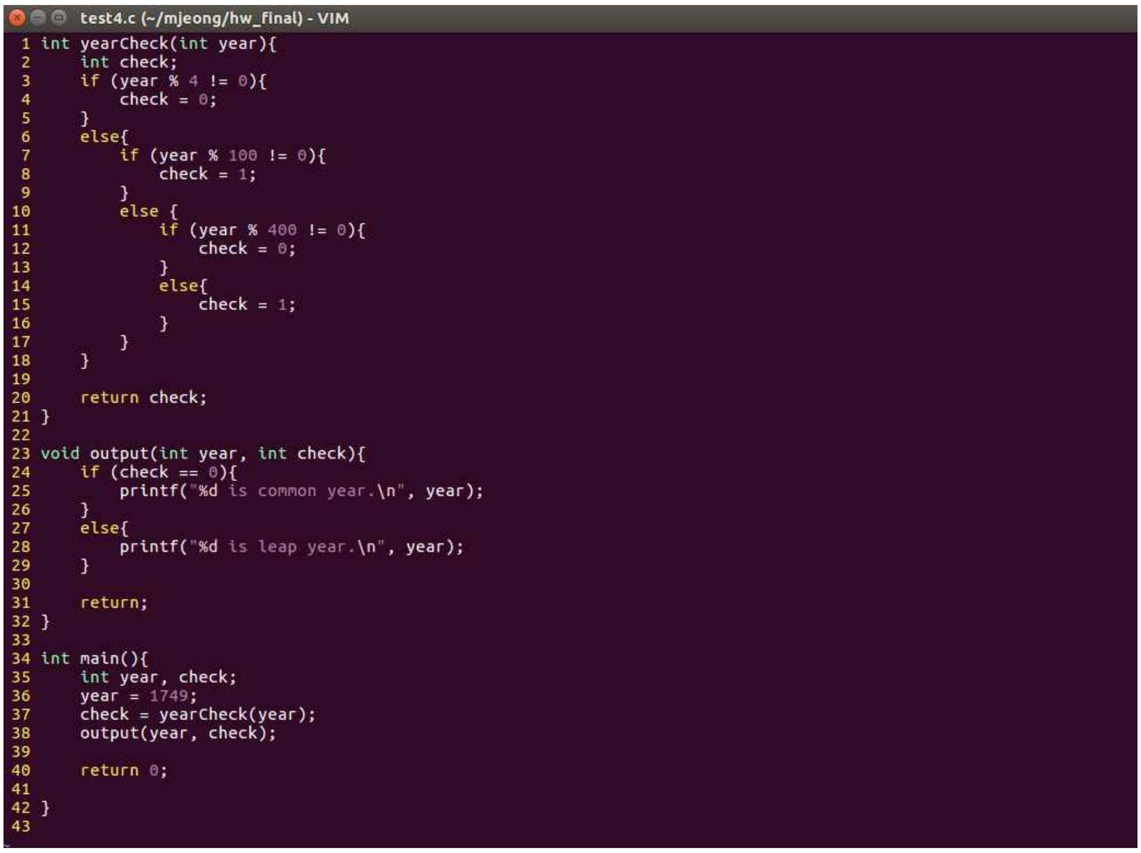
test3.c를 gcc로 실행하면 다음과 같은 결과가 나온다. 두 개가 동일한 결과라는 것을 확인할 수 있다.



```
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

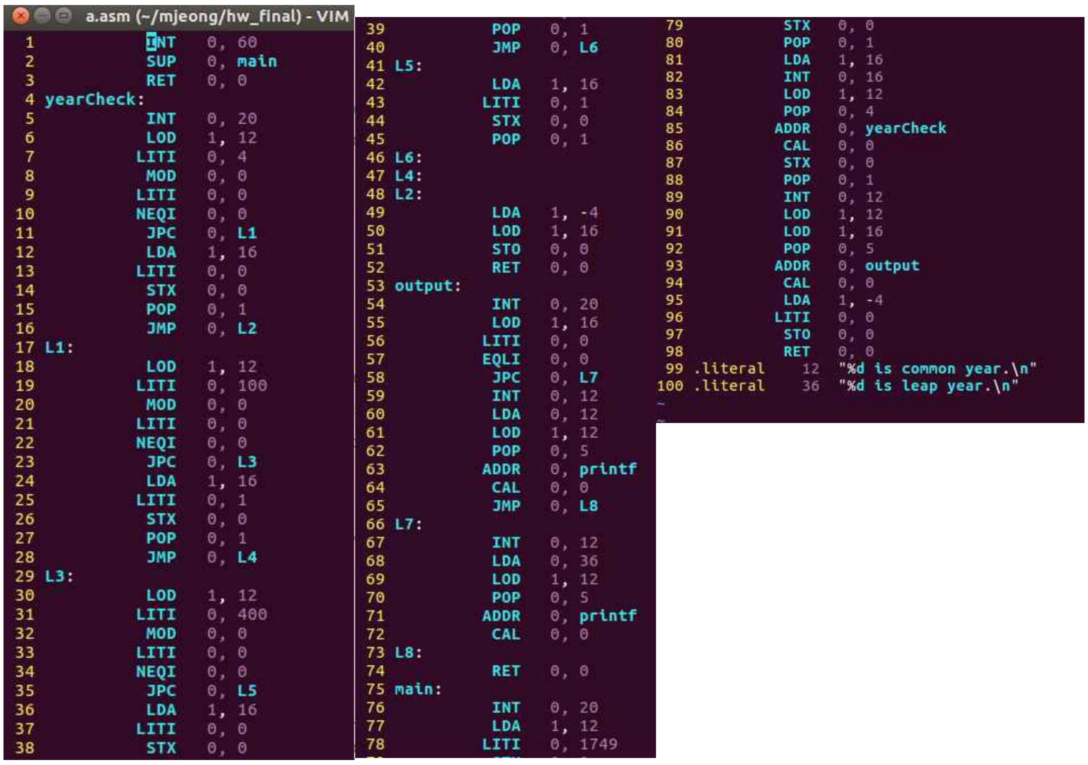
윤년인지 평년인지 판단하는 프로그램이다.



test4.c (~/mjeong/hw_final) - VIM

```c
int yearCheck(int year){
    int check;
    if (year % 4 != 0){
        check = 0;
    }
    else{
        if (year % 100 != 0){
            check = 1;
        }
        else {
            if (year % 400 != 0){
                check = 0;
            }
            else{
                check = 1;
            }
        }
    }

    return check;
}

void output(int year, int check){
    if (check == 0){
        printf("%d is common year.\n", year);
    }
    else{
        printf("%d is leap year.\n", year);
    }

    return;
}

int main(){
    int year, check;
    year = 1749;
    check = yearCheck(year);
    output(year, check);

    return 0;

}
```

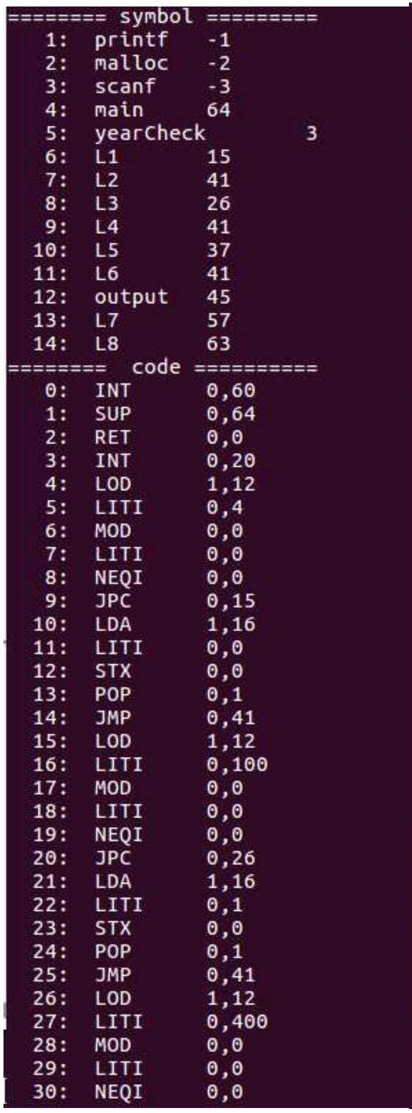
● 컴파일러를 통하여 만들어진 어셈블러 프로그램은 다음과 같다.



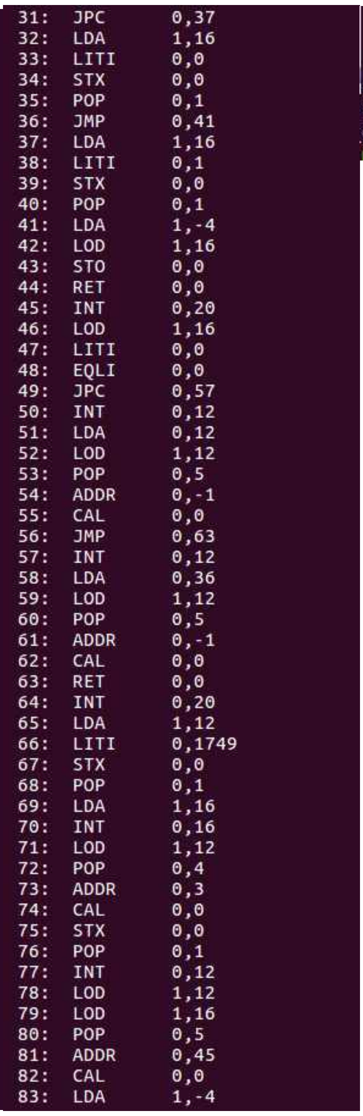
a.asm (~/mjeong/hw_final) - VIM

```
                INT     0, 60
                SUP     0, main
                RET     0, 0
yearCheck:
                INT     0, 20
                LOD     1, 12
               LITI     0, 4
                MOD     0, 0
               LITI     0, 0
               NEQI     0, 0
                JPC     0, L1
                LDA     1, 16
               LITI     0, 0
                STX     0, 0
                POP     0, 1
                JMP     0, L2
L1:
                LOD     1, 12
               LITI     0, 100
                MOD     0, 0
               LITI     0, 0
               NEQI     0, 0
                JPC     0, L3
                LDA     1, 16
               LITI     0, 1
                STX     0, 0
                POP     0, 1
                JMP     0, L4
L3:
                LOD     1, 12
               LITI     0, 400
                MOD     0, 0
               LITI     0, 0
               NEQI     0, 0
                JPC     0, L5
                LDA     1, 16
               LITI     0, 0
                STX     0, 0
                POP     0, 1
                JMP     0, L6
L5:
                LDA     1, 16
               LITI     0, 1
                STX     0, 0
                POP     0, 1
L6:
L4:
L2:
                LDA     1, -4
                LOD     1, 16
                STO     0, 0
                RET     0, 0
output:
                INT     0, 20
                LOD     1, 16
               LITI     0, 0
               EQLI     0, 0
                JPC     0, L7
                INT     0, 12
                LDA     0, 12
                LOD     1, 12
                POP     0, 5
               ADDR     0, printf
                CAL     0, 0
                JMP     0, L8
L7:
                INT     0, 12
                LDA     0, 36
                LOD     1, 12
                POP     0, 5
               ADDR     0, printf
                CAL     0, 0
L8:
                RET     0, 0
main:
                INT     0, 20
                LDA     1, 12
               LITI     0, 1749
                STX     0, 0
                POP     0, 1
                LDA     1, 16
                INT     0, 16
                LOD     1, 12
                POP     0, 4
               ADDR     0, yearCheck
                CAL     0, 0
                STX     0, 0
                POP     0, 1
                INT     0, 12
                LOD     1, 12
                LOD     1, 16
                POP     0, 5
               ADDR     0, output
                CAL     0, 0
                LDA     1, -4
               LITI     0, 0
                STO     0, 0
                RET     0, 0
.literal     12   "%d is common year.\n"
.literal     36   "%d is leap year.\n"
```

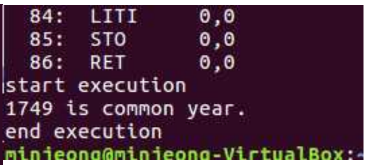
● 어셈블러 프로그램을 시뮬레이터를 이용하여 실행하면 다음과 같은 결과가 나온다.



```
======== 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
```

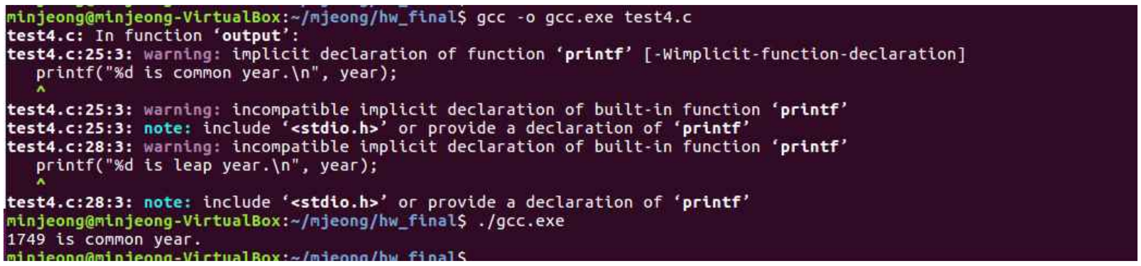


```
  31:  JPC       0,37
  32:  LDA       1,16
  33:  LITI      0,0
  34:  STX       0,0
  35:  POP       0,1
  36:  JMP       0,41
  37:  LDA       1,16
  38:  LITI      0,1
  39:  STX       0,0
  40:  POP       0,1
  41:  LDA       1,-4
  42:  LOD       1,16
  43:  STO       0,0
  44:  RET       0,0
  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
```



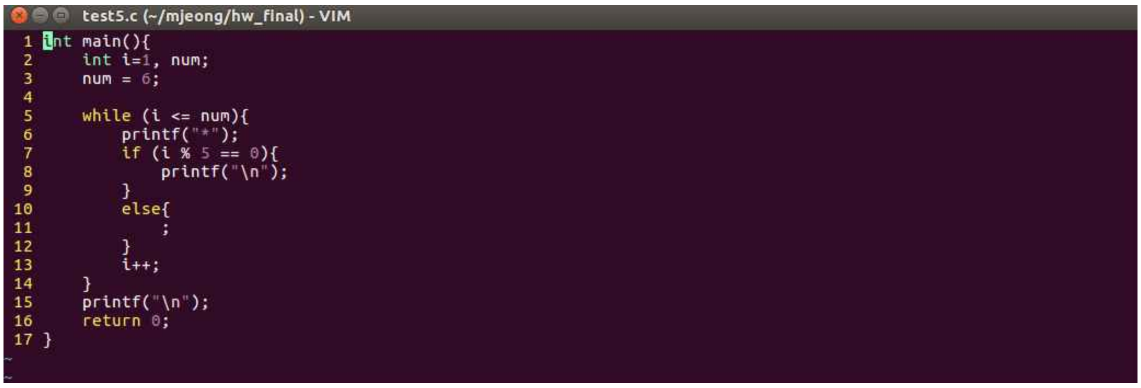
```
  84:  LITI      0,0
  85:  STO       0,0
  86:  RET       0,0
start execution
1749 is common year.
end execution
minjeong@minjeong-VirtualBox:~
```

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



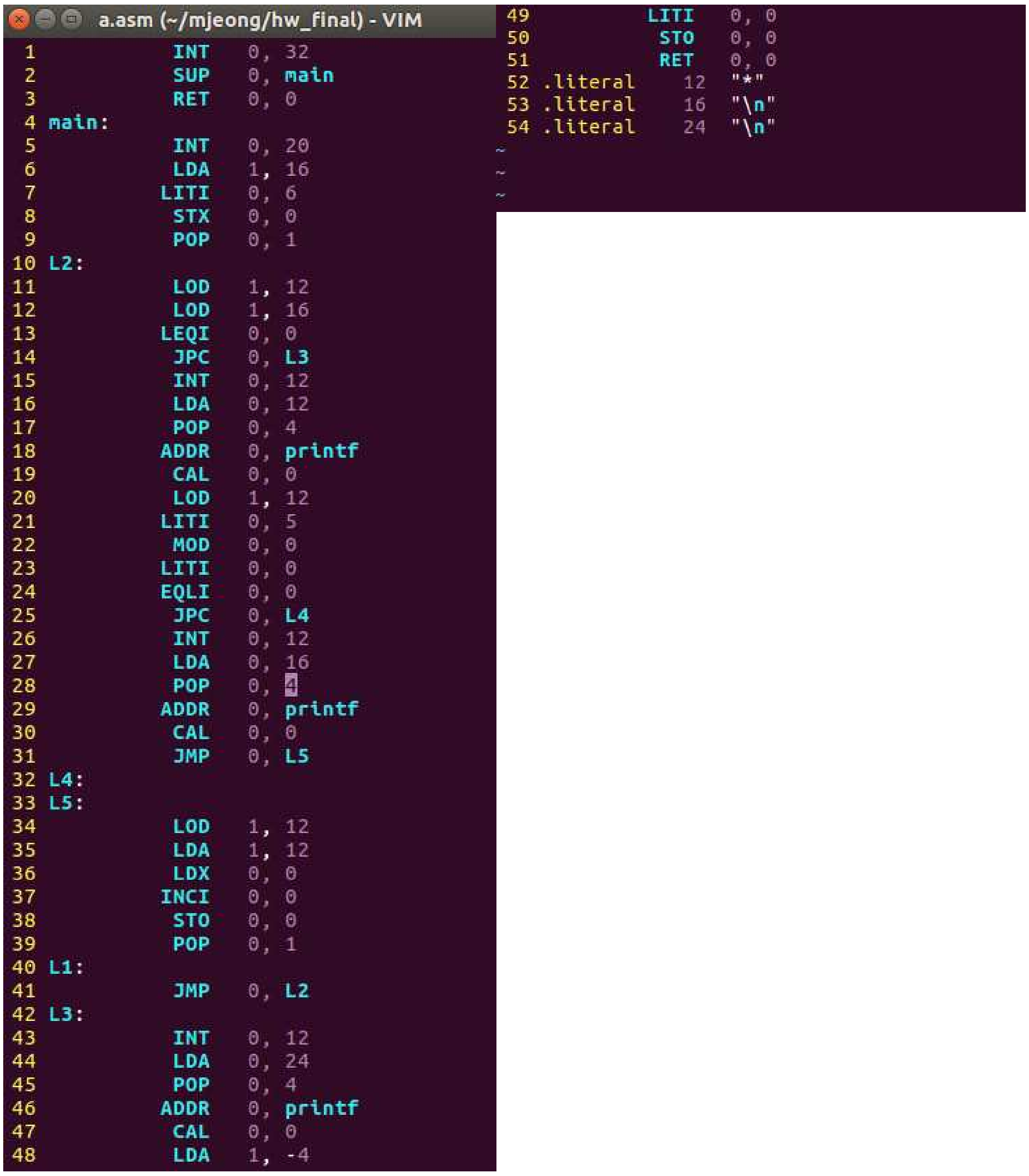
```
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$ gcc -o gcc.exe test4.c
test4.c: In function ‘output’:
test4.c:25:3: warning: implicit declaration of function ‘printf’ [-Wimplicit-function-declaration]
   printf("%d is common year.\n", year);
   ^
test4.c:25:3: warning: incompatible implicit declaration 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:~/mjeong/hw_final$ ./gcc.exe
1749 is common year.
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$
```

5) test5.c



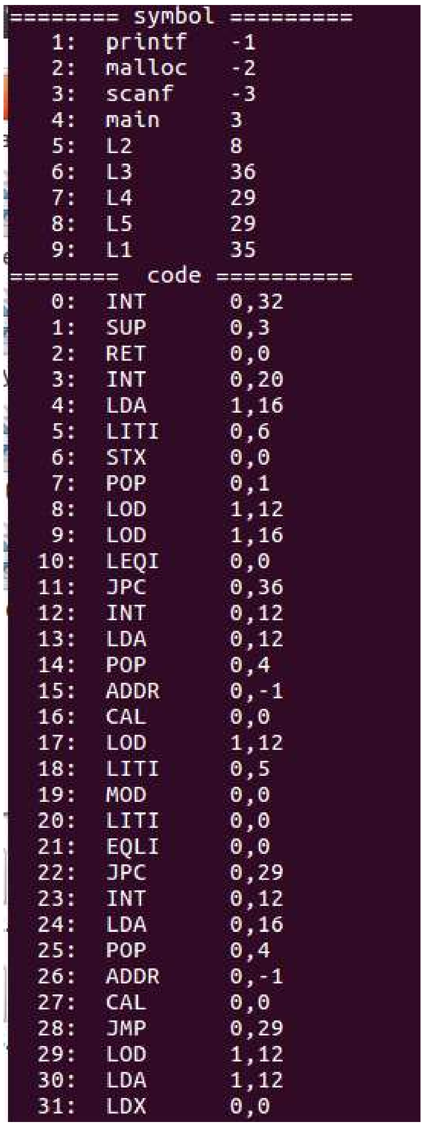
```
int main(){
    int i=1, num;
    num = 6;

    while (i <= num){
        printf("*");
        if (i % 5 == 0){
            printf("\n");
        }
        else{
            ;
        }
        i++;
    }
    printf("\n");
    return 0;
}
```

● 컴파일러를 통하여 만들어진 어셈블러 프로그램은 다음과 같다.

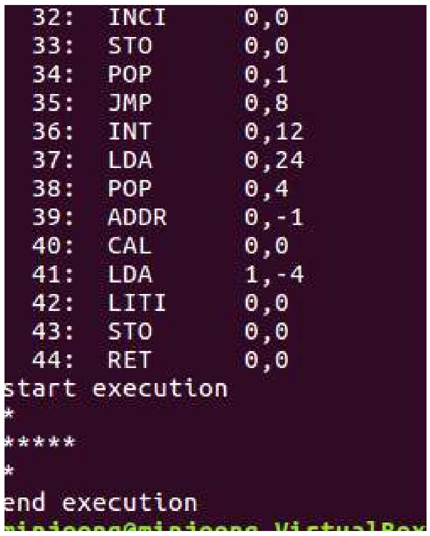


```
              INT    0, 32
              SUP    0, main
              RET    0, 0
main:
              INT    0, 20
              LDA    1, 16
             LITI    0, 6
              STX    0, 0
              POP    0, 1
L2:
              LOD    1, 12
              LOD    1, 16
             LEQI    0, 0
              JPC    0, L3
              INT    0, 12
              LDA    0, 12
              POP    0, 4
             ADDR    0, printf
              CAL    0, 0
              LOD    1, 12
             LITI    0, 5
              MOD    0, 0
             LITI    0, 0
             EQLI    0, 0
              JPC    0, L4
              INT    0, 12
              LDA    0, 16
              POP    0, 4
             ADDR    0, printf
              CAL    0, 0
              JMP    0, L5
L4:
L5:
              LOD    1, 12
              LDA    1, 12
              LDX    0, 0
             INCI    0, 0
              STO    0, 0
              POP    0, 1
L1:
              JMP    0, L2
L3:
              INT    0, 12
              LDA    0, 24
              POP    0, 4
             ADDR    0, printf
              CAL    0, 0
              LDA    1, -4
             LITI    0, 0
              STO    0, 0
              RET    0, 0
.literal     12    "*"
.literal     16    "\n"
.literal     24    "\n"
```

● 어셈블러 프로그램을 시뮬레이터를 이용하여 실행하면 다음과 같은 결과가 나온다.

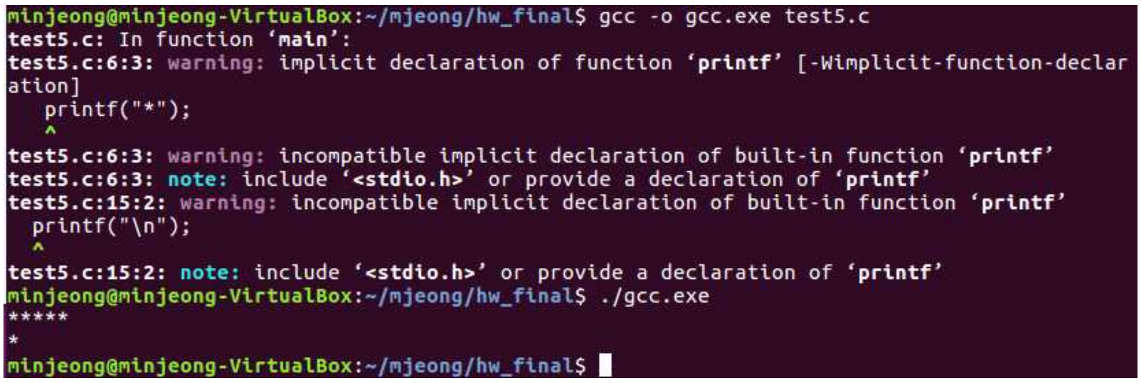


```
======== symbol =========
   1:  printf   -1
   2:  malloc   -2
   3:  scanf    -3
   4:  main     3
   5:  L2       8
   6:  L3       36
   7:  L4       29
   8:  L5       29
   9:  L1       35
========  code ==========
   0:  INT      0,32
   1:  SUP      0,3
   2:  RET      0,0
   3:  INT      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
```

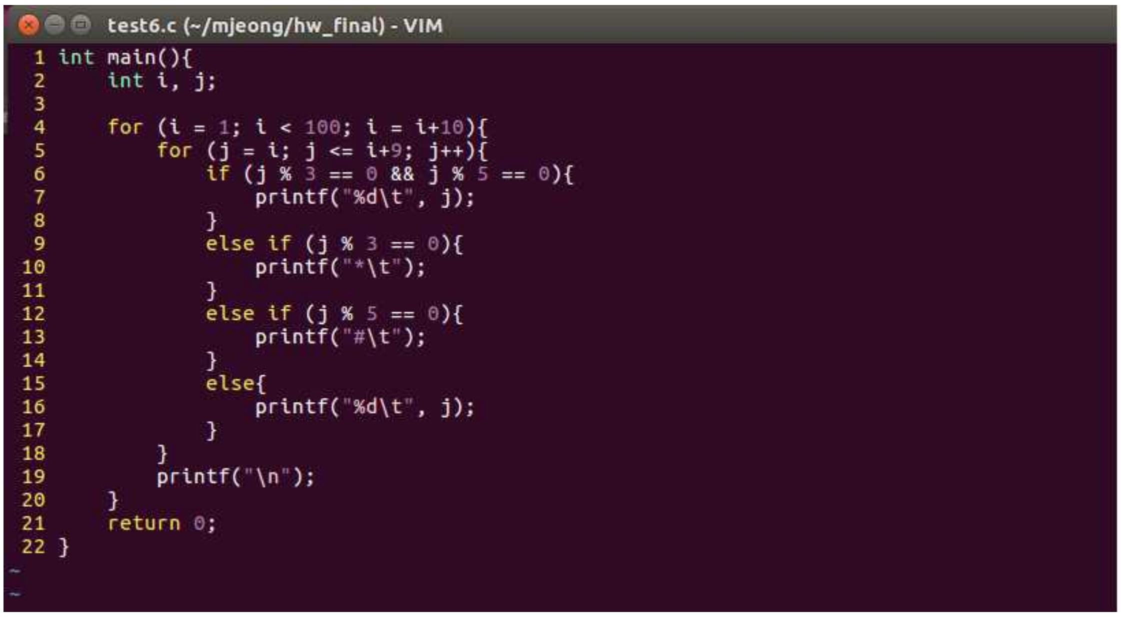
test5.c를 gcc로 실행하면 다음과 같은 결과가 나온다. 예상과 달리 시뮬레이터를 이용하여 실행한 결과가 이상하게 출력되었다. 그 원인에 대하여 고찰이 필요할 것같다.



```
minjeong@minjeong-VirtualBox:~/mjeong/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-declar
ation]
   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:~/mjeong/hw_final$ ./gcc.exe
*****
*
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$ 
```

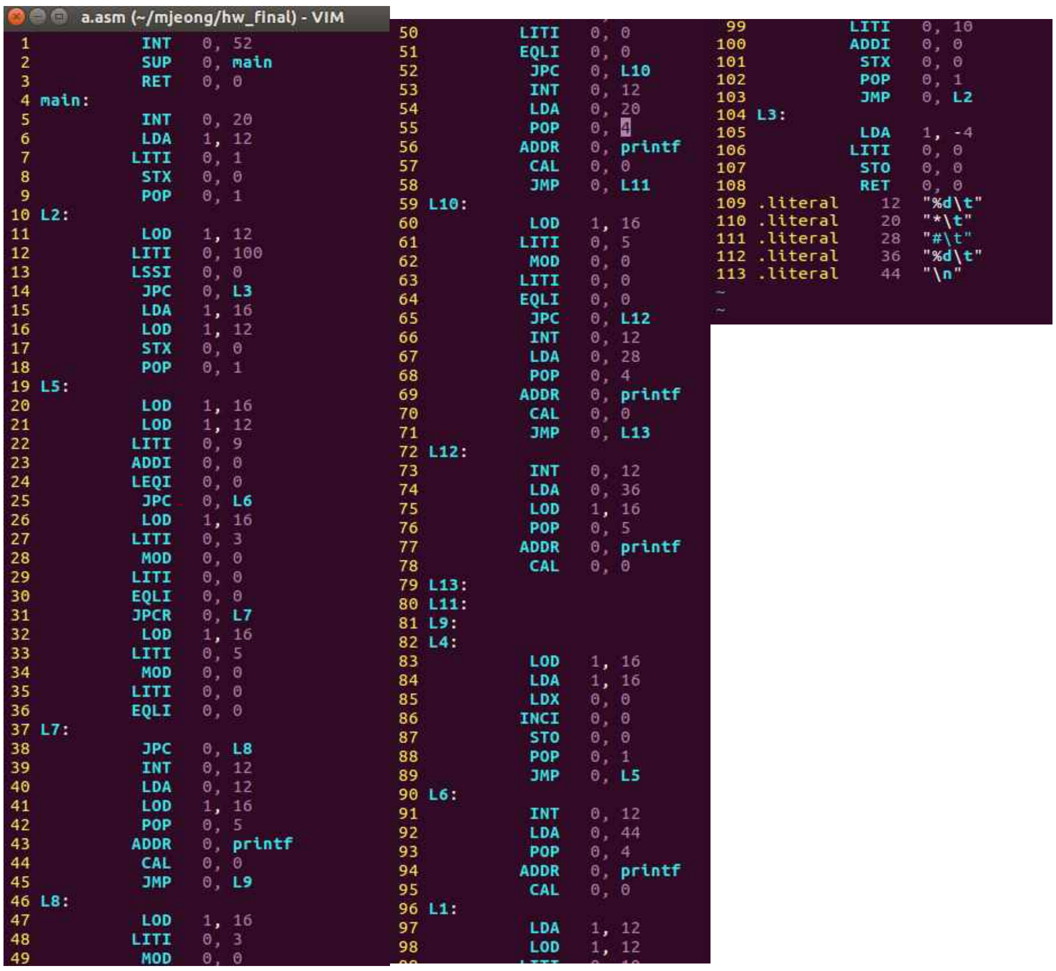
6) test6.c

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

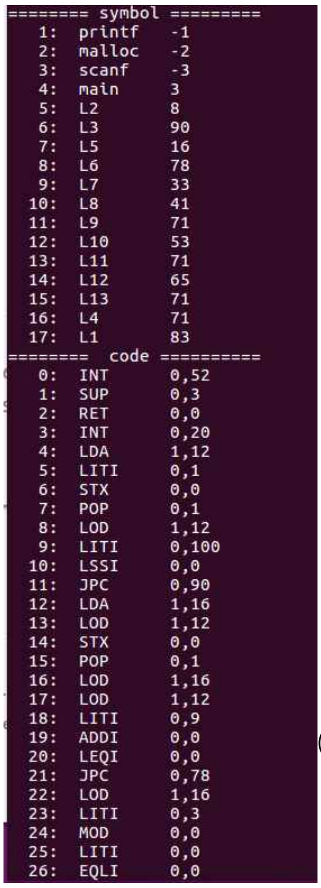
```c
int main(){
    int i, j;

    for (i = 1; i < 100; i = i+10){
        for (j = i; j <= i+9; j++){
            if (j % 3 == 0 && j % 5 == 0){
                printf("%d\t", j);
            }
            else if (j % 3 == 0){
                printf("*\t");
            }
            else if (j % 5 == 0){
                printf("#\t");
            }
            else{
                printf("%d\t", j);
            }
        }
        printf("\n");
    }
    return 0;
}
```

● 컴파일러를 통하여 만들어진 어셈블러 프로그램은 다음과 같다.


a.asm (~/mjeong/hw_final) - VIM

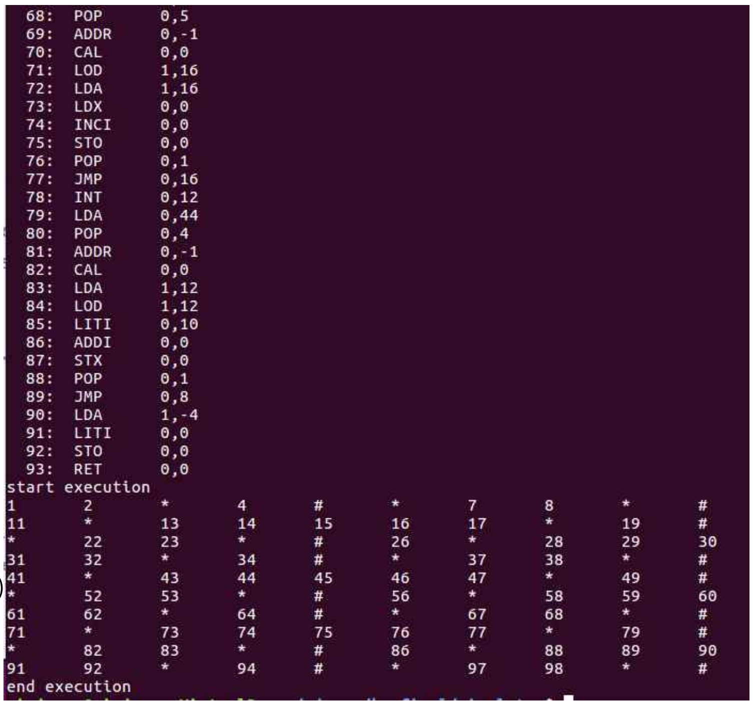
```
                INT     0, 52
                SUP     0, main
                RET     0, 0
main:
                INT     0, 20
                LDA     1, 12
                LITI    0, 1
                STX     0, 0
                POP     0, 1
L2:
                LOD     1, 12
                LITI    0, 100
                LSSI    0, 0
                JPC     0, L3
                LDA     1, 16
                LOD     1, 12
                STX     0, 0
                POP     0, 1
L5:
                LOD     1, 16
                LOD     1, 12
                LITI    0, 9
                ADDI    0, 0
                LEQI    0, 0
                JPC     0, L6
                LOD     1, 16
                LITI    0, 3
                MOD     0, 0
                LITI    0, 0
                EQLI    0, 0
                JPCR    0, L7
                LOD     1, 16
                LITI    0, 5
                MOD     0, 0
                LITI    0, 0
                EQLI    0, 0
L7:
                JPC     0, L8
                INT     0, 12
                LDA     0, 12
                LOD     1, 16
                POP     0, 5
                ADDR    0, printf
                CAL     0, 0
                JMP     0, L9
L8:
                LOD     1, 16
                LITI    0, 3
                MOD     0, 0
                LITI    0, 0
                EQLI    0, 0
                JPC     0, L10
                INT     0, 12
                LDA     0, 20
                POP     0, 4
                ADDR    0, printf
                CAL     0, 0
                JMP     0, L11
L10:
                LOD     1, 16
                LITI    0, 5
                MOD     0, 0
                LITI    0, 0
                EQLI    0, 0
                JPC     0, L12
                INT     0, 12
                LDA     0, 28
                POP     0, 4
                ADDR    0, printf
                CAL     0, 0
                JMP     0, L13
L12:
                INT     0, 12
                LDA     0, 36
                LOD     1, 16
                POP     0, 5
                ADDR    0, printf
                CAL     0, 0
L13:
L11:
L9:
L4:
                LOD     1, 16
                LDA     1, 16
                LDX     0, 0
                INCI    0, 0
                STO     0, 0
                POP     0, 1
                JMP     0, L5
L6:
                INT     0, 12
                LDA     0, 44
                POP     0, 4
                ADDR    0, printf
                CAL     0, 0
L1:
                LDA     1, 12
                LOD     1, 12
                LITI    0, 10
                ADDI    0, 0
                STX     0, 0
                POP     0, 1
                JMP     0, L2
L3:
                LDA     1, -4
                LITI    0, 0
                STO     0, 0
                RET     0, 0
.literal        12      "%d\t"
.literal        20      "*\t"
.literal        28      "#\t"
.literal        36      "%d\t"
.literal        44      "\n"
```

● 어셈블러 프로그램을 시뮬레이터를 이용하여 실행하면 다음과 같은 결과가 나온다.



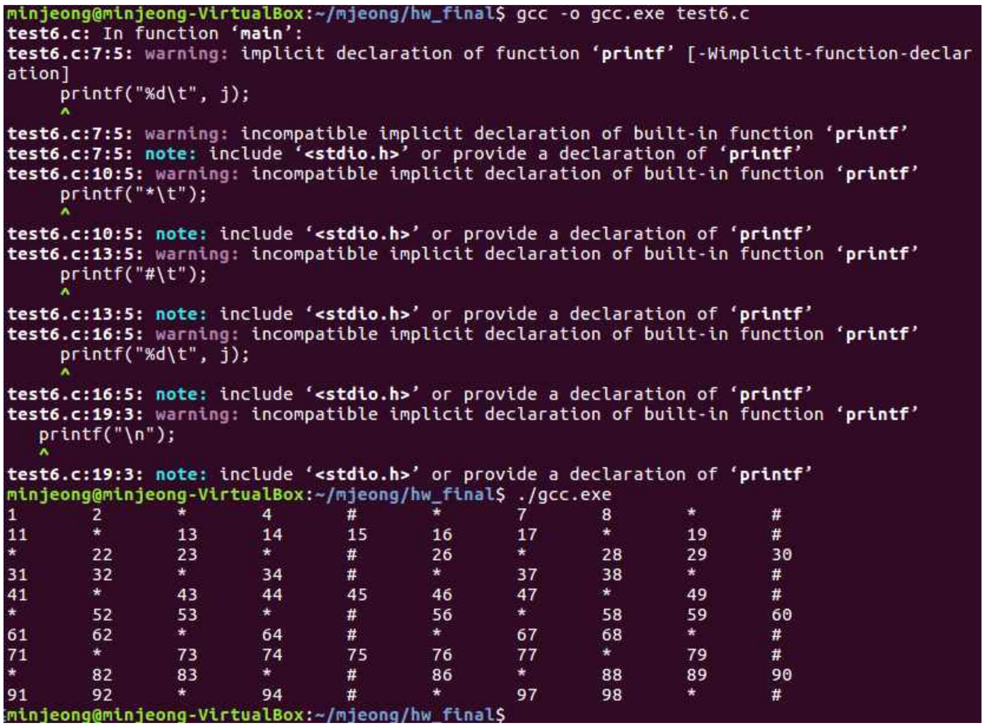
```
======== symbol =========
    1:  printf    -1
    2:  malloc    -2
    3:  scanf     -3
    4:  main      3
    5:  L2        8
    6:  L3        90
    7:  L5        16
    8:  L6        78
    9:  L7        33
   10:  L8        41
   11:  L9        71
   12:  L10       53
   13:  L11       71
   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
```

(생략)



```
  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
```

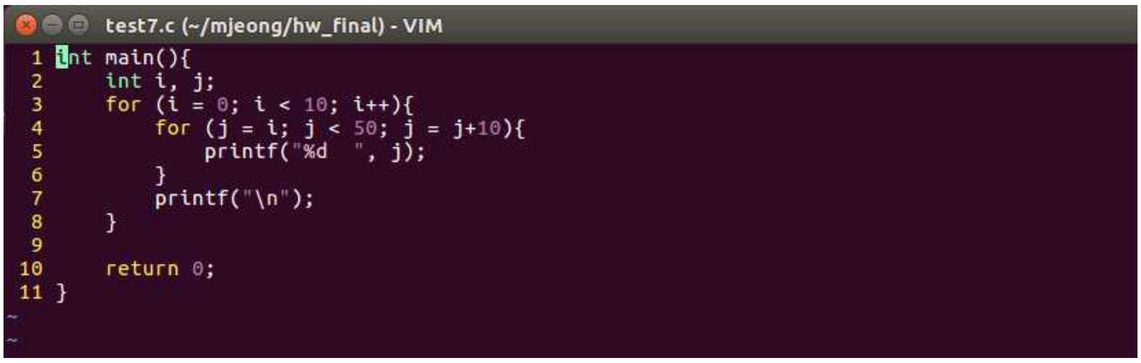
test6.c를 gcc로 실행하면 다음과 같은 결과가 나온다. 두 개가 동일한 결과라는 것을 확인할 수 있다.



```
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-declar
ation]
     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");
     ^
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");
     ^
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$
```

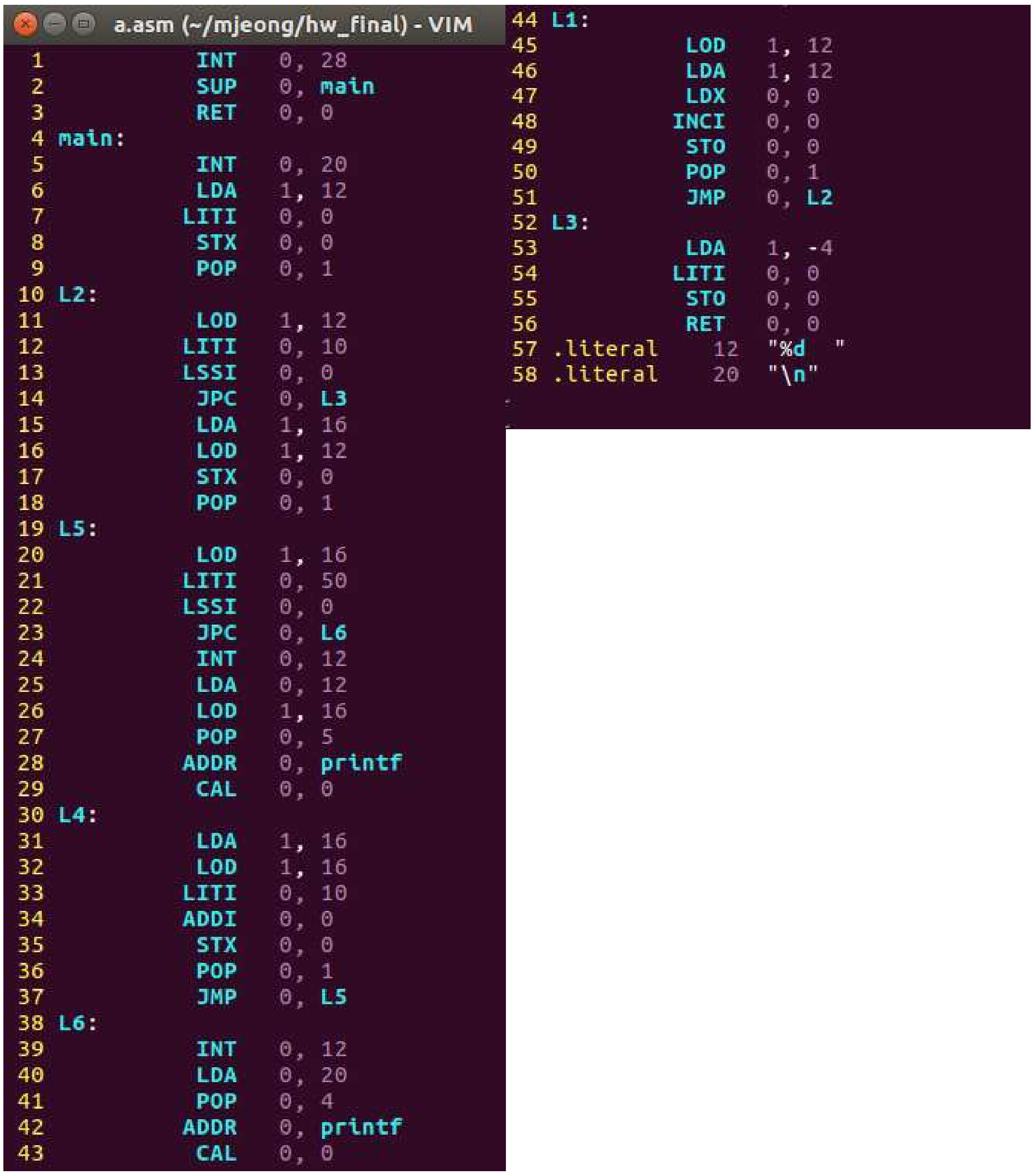
7) test7.c

0부터 49까지의 숫자를 10개단위로 나눠서 출력하는 프로그램이다.



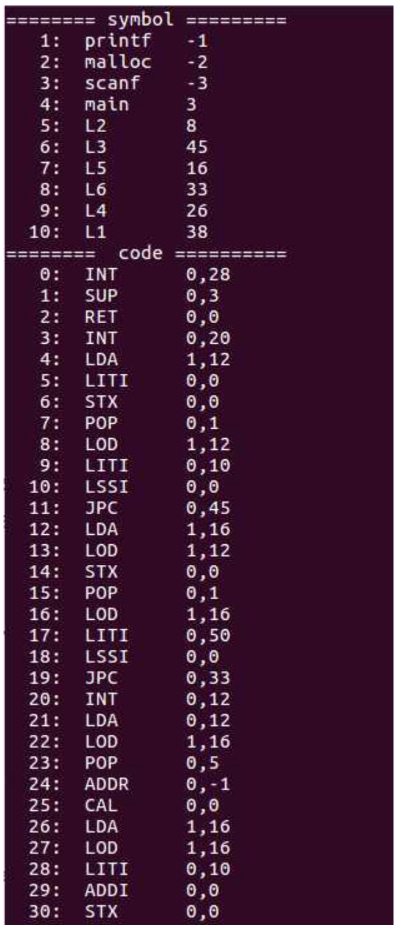
```
int main(){
    int i, j;
    for (i = 0; i < 10; i++){
        for (j = i; j < 50; j = j+10){
            printf("%d  ", j);
        }
        printf("\n");
    }

    return 0;
}
```

● 컴파일러를 통하여 만들어진 어셈블러 프로그램은 다음과 같다.

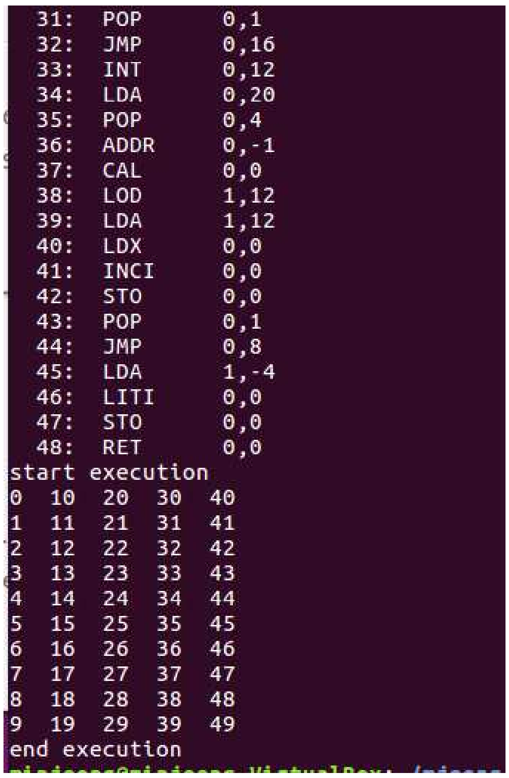


```
            INT    0, 28
            SUP    0, main
            RET    0, 0
main:
            INT    0, 20
            LDA    1, 12
            LITI   0, 0
            STX    0, 0
            POP    0, 1
L2:
            LOD    1, 12
            LITI   0, 10
            LSSI   0, 0
            JPC    0, L3
            LDA    1, 16
            LOD    1, 12
            STX    0, 0
            POP    0, 1
L5:
            LOD    1, 16
            LITI   0, 50
            LSSI   0, 0
            JPC    0, L6
            INT    0, 12
            LDA    0, 12
            LOD    1, 16
            POP    0, 5
            ADDR   0, printf
            CAL    0, 0
L4:
            LDA    1, 16
            LOD    1, 16
            LITI   0, 10
            ADDI   0, 0
            STX    0, 0
            POP    0, 1
            JMP    0, L5
L6:
            INT    0, 12
            LDA    0, 20
            POP    0, 4
            ADDR   0, printf
            CAL    0, 0
L1:
            LOD    1, 12
            LDA    1, 12
            LDX    0, 0
            INCI   0, 0
            STO    0, 0
            POP    0, 1
            JMP    0, L2
L3:
            LDA    1, -4
            LITI   0, 0
            STO    0, 0
            RET    0, 0
.literal      12   "%d  "
.literal      20   "\n"
```

● 어셈블러 프로그램을 시뮬레이터를 이용하여 실행하면 다음과 같은 결과가 나온다.

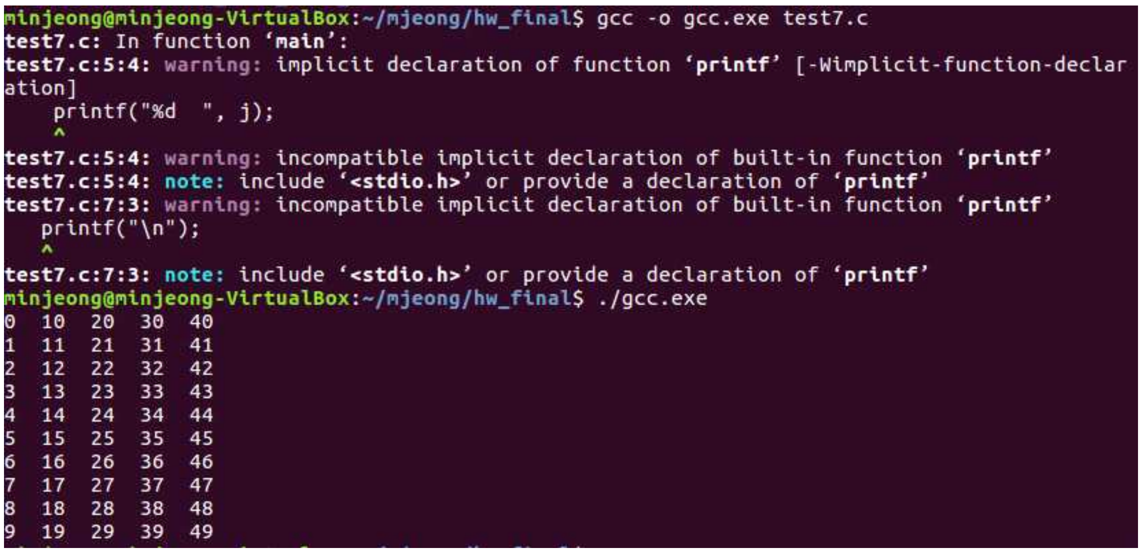


```
======== symbol =========
   1:  printf   -1
   2:  malloc   -2
   3:  scanf    -3
   4:  main     3
   5:  L2       8
   6:  L3       45
   7:  L5       16
   8:  L6       33
   9:  L4       26
  10:  L1       38
========  code ==========
   0:  INT      0,28
   1:  SUP      0,3
   2:  RET      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
```

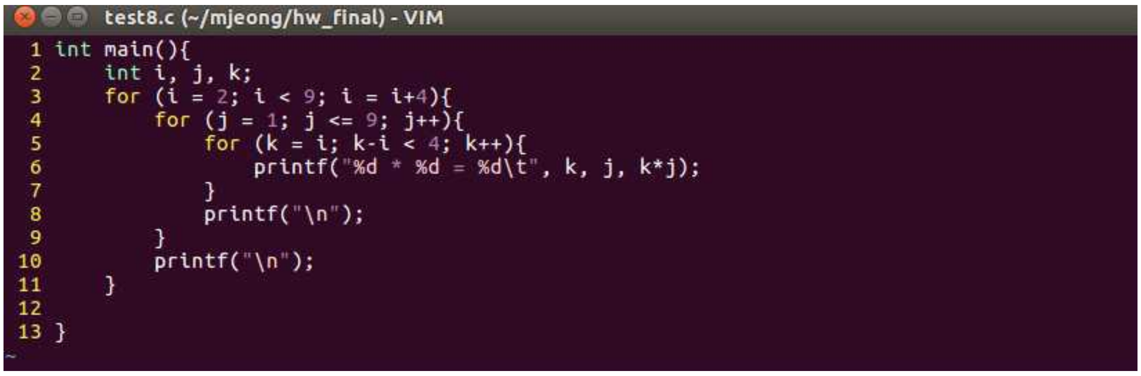
test7.c를 gcc로 실행하면 다음과 같은 결과가 나온다. 두 개가 동일한 결과라는 것을 확인할 수 있다.



```
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-declar
ation]
    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

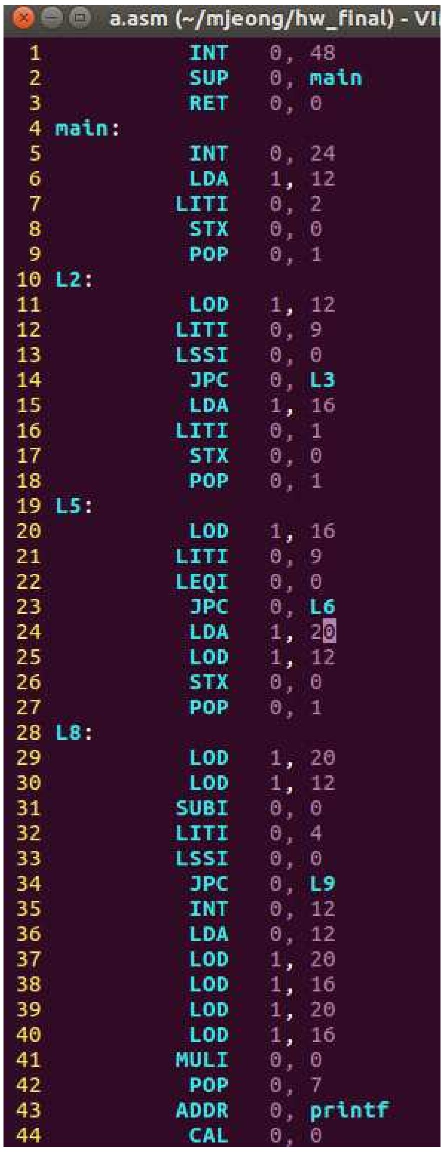
삼중 for문을 이용하여 구구단을 출력한 프로그램이다.



test8.c (~/mjeong/hw_final) - VIM

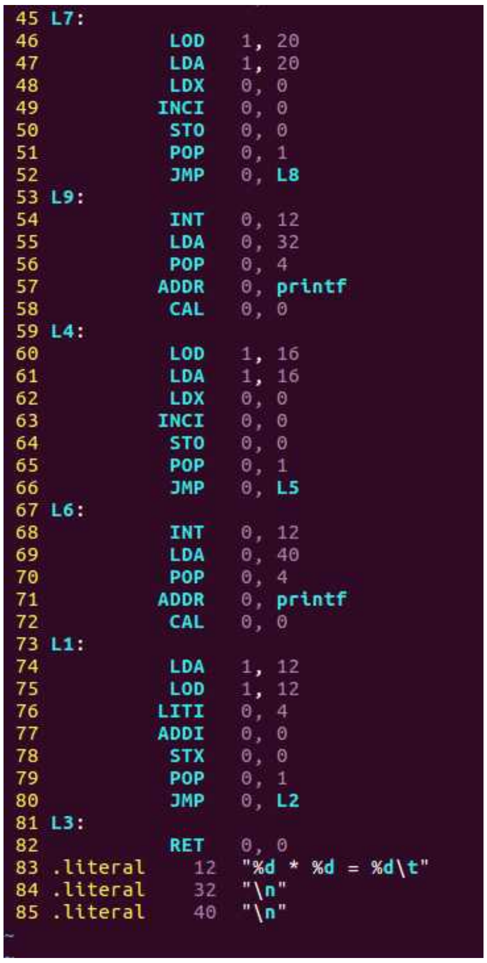
```
int main(){
    int i, j, k;
    for (i = 2; i < 9; i = i+4){
        for (j = 1; j <= 9; j++){
            for (k = i; k-i < 4; k++){
                printf("%d * %d = %d\t", k, j, k*j);
            }
            printf("\n");
        }
        printf("\n");
    }

}
```

● 컴파일러를 통하여 만들어진 어셈블러 프로그램은 다음과 같다.



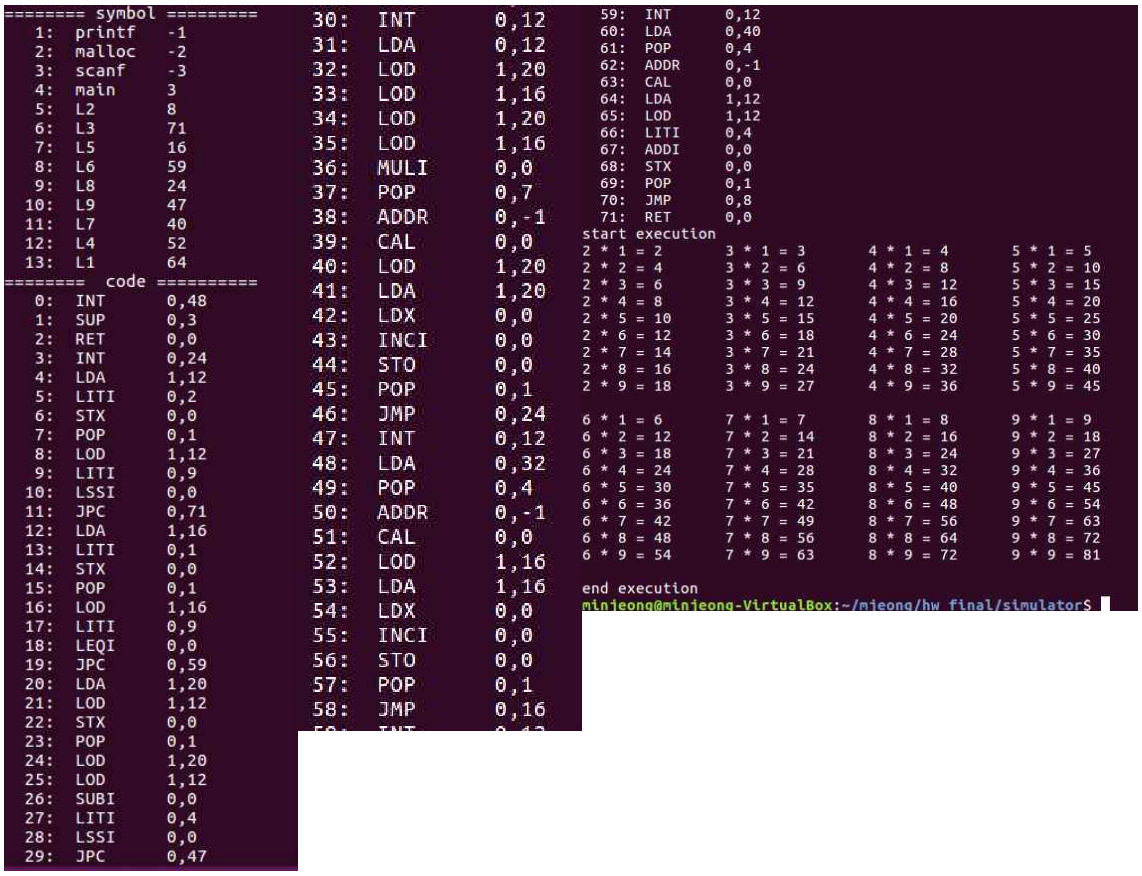
a.asm (~/mjeong/hw_final) - VI

```
          INT    0, 48
          SUP    0, main
          RET    0, 0
main:
          INT    0, 24
          LDA    1, 12
          LITI   0, 2
          STX    0, 0
          POP    0, 1
L2:
          LOD    1, 12
          LITI   0, 9
          LSSI   0, 0
          JPC    0, L3
          LDA    1, 16
          LITI   0, 1
          STX    0, 0
          POP    0, 1
L5:
          LOD    1, 16
          LITI   0, 9
          LEQI   0, 0
          JPC    0, L6
          LDA    1, 20
          LOD    1, 12
          STX    0, 0
          POP    0, 1
L8:
          LOD    1, 20
          LOD    1, 12
          SUBI   0, 0
          LITI   0, 4
          LSSI   0, 0
          JPC    0, L9
          INT    0, 12
          LDA    0, 12
          LOD    1, 20
          LOD    1, 16
          LOD    1, 20
          LOD    1, 16
          MULI   0, 0
          POP    0, 7
          ADDR   0, printf
          CAL    0, 0
```



```
L7:
          LOD    1, 20
          LDA    1, 20
          LDX    0, 0
          INCI   0, 0
          STO    0, 0
          POP    0, 1
          JMP    0, L8
L9:
          INT    0, 12
          LDA    0, 32
          POP    0, 4
          ADDR   0, printf
          CAL    0, 0
L4:
          LOD    1, 16
          LDA    1, 16
          LDX    0, 0
          INCI   0, 0
          STO    0, 0
          POP    0, 1
          JMP    0, L5
L6:
          INT    0, 12
          LDA    0, 40
          POP    0, 4
          ADDR   0, printf
          CAL    0, 0
L1:
          LDA    1, 12
          LOD    1, 12
          LITI   0, 4
          ADDI   0, 0
          STX    0, 0
          POP    0, 1
          JMP    0, L2
L3:
          RET    0, 0
.literal    12   "%d * %d = %d\t"
.literal    32   "\n"
.literal    40   "\n"
```

● 어셈블러 프로그램을 시뮬레이터를 이용하여 실행하면 다음과 같은 결과가 나온다.

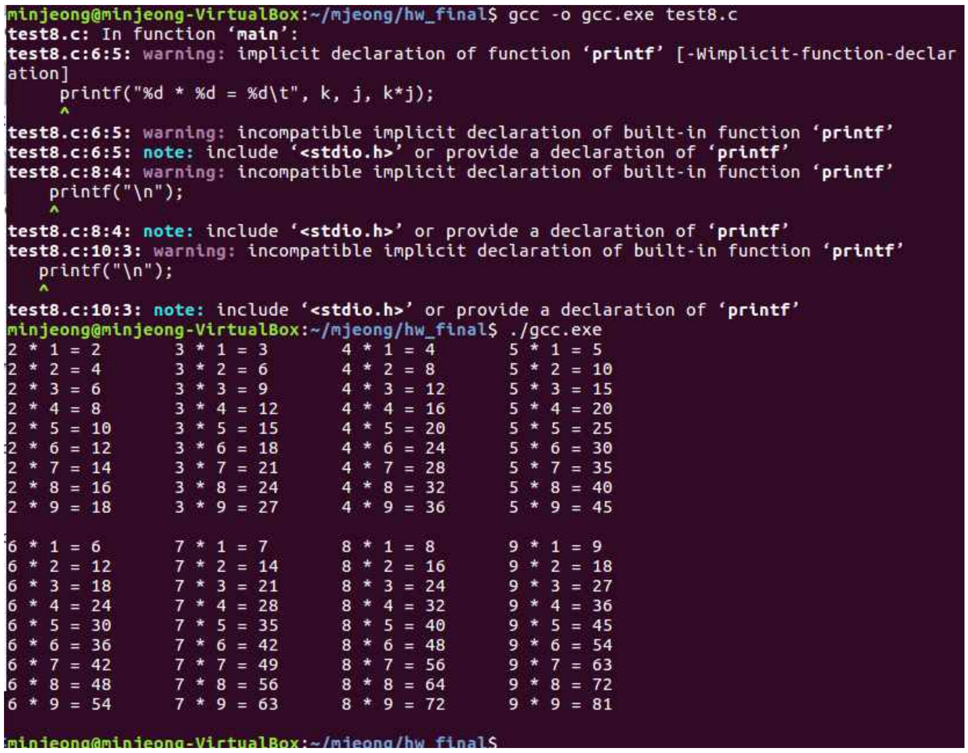


```
======== symbol =========
   1:  printf    -1
   2:  malloc    -2
   3:  scanf     -3
   4:  main      3
   5:  L2        8
   6:  L3        71
   7:  L5        16
   8:  L6        59
   9:  L8        24
  10:  L9        47
  11:  L7        40
  12:  L4        52
  13:  L1        64
=======   code  =========
   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:~/mjeong/hw_final/simulator$
```

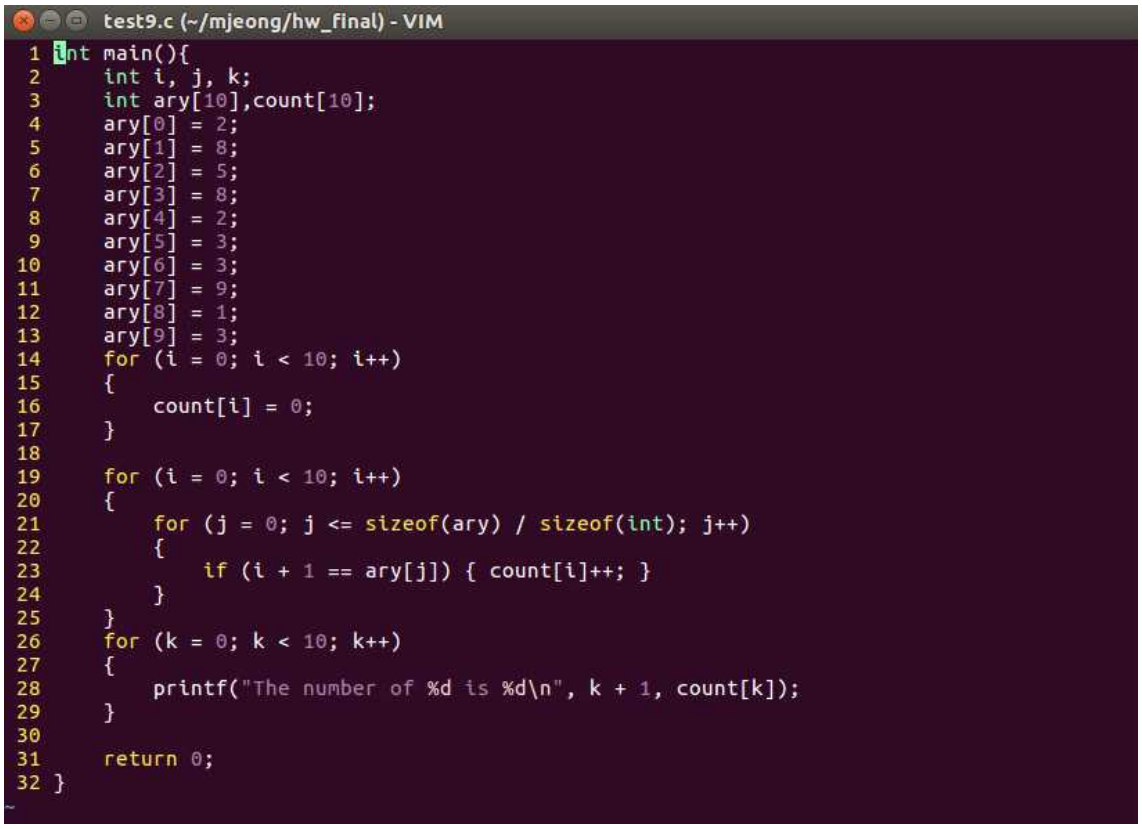
test8.c를 gcc로 실행하면 다음과 같은 결과가 나온다. 두 개가 동일한 결과라는 것을 확인할 수 있다.



```
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$ gcc -o gcc.exe test8.c
test8.c: In function ‘main’:
test8.c:6:5: warning: implicit declaration of function ‘printf’ [-Wimplicit-function-declar
ation]
     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:~/mjeong/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:~/mjeong/hw_final$
```

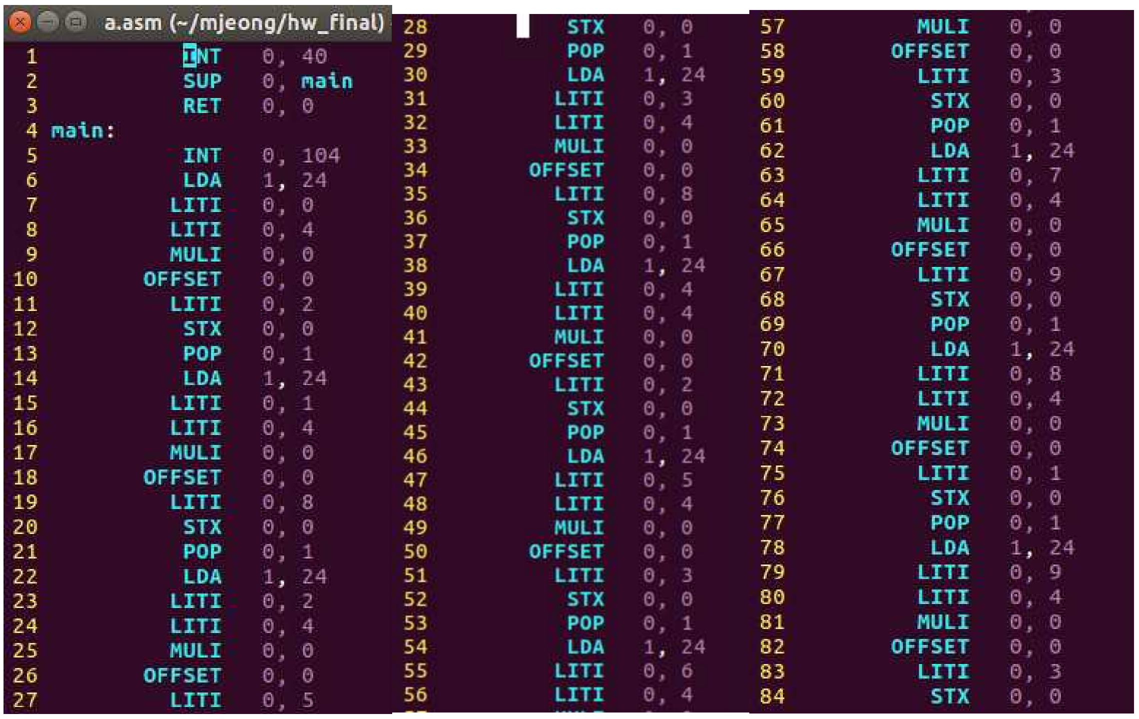
9) test9.c

배열 안에 있는 숫자의 개수를 세는 프로그램이다.

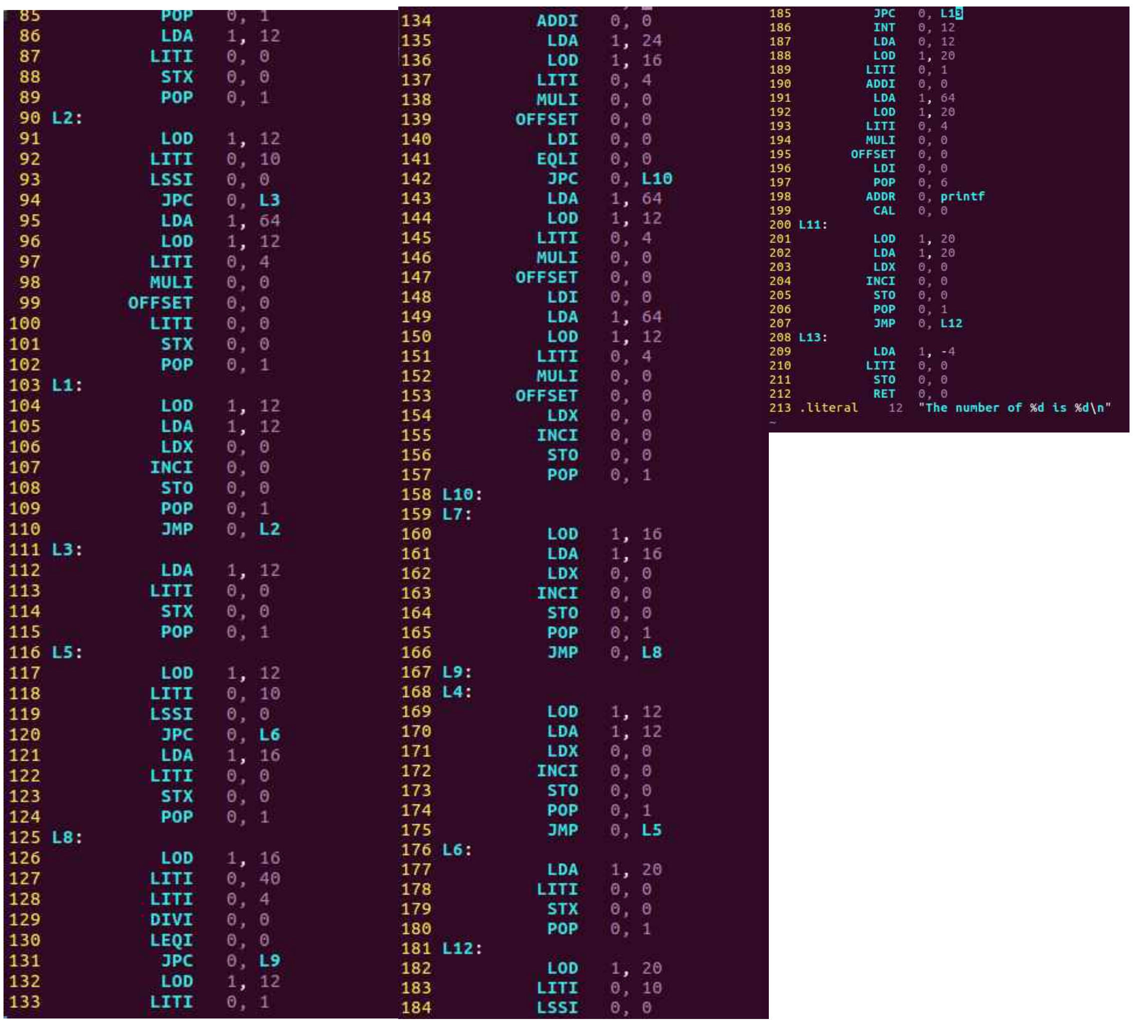

test9.c (~/mjeong/hw_final) - VIM

```c
int main(){
    int i, j, k;
    int ary[10],count[10];
    ary[0] = 2;
    ary[1] = 8;
    ary[2] = 5;
    ary[3] = 8;
    ary[4] = 2;
    ary[5] = 3;
    ary[6] = 3;
    ary[7] = 9;
    ary[8] = 1;
    ary[9] = 3;
    for (i = 0; i < 10; i++)
    {
        count[i] = 0;
    }

    for (i = 0; i < 10; i++)
    {
        for (j = 0; j <= sizeof(ary) / sizeof(int); j++)
        {
            if (i + 1 == ary[j]) { count[i]++; }
        }
    }
    for (k = 0; k < 10; k++)
    {
        printf("The number of %d is %d\n", k + 1, count[k]);
    }

    return 0;
}
```

● 컴파일러를 통하여 만들어진 어셈블러 프로그램은 다음과 같다.

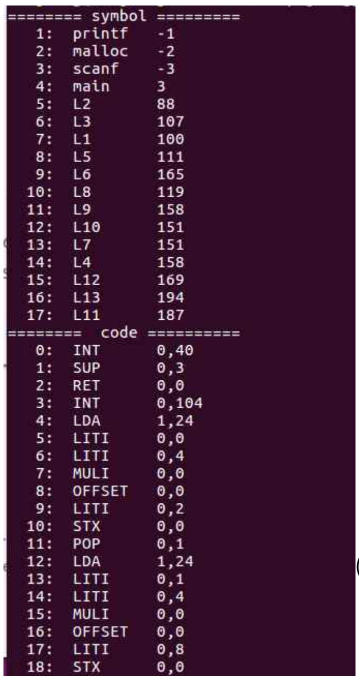

a.asm (~/mjeong/hw_final)

```
          INT     0, 40
          SUP     0, main
          RET     0, 0
main:
          INT     0, 104
          LDA     1, 24
          LITI    0, 0
          LITI    0, 4
          MULI    0, 0
          OFFSET  0, 0
          LITI    0, 2
          STX     0, 0
          POP     0, 1
          LDA     1, 24
          LITI    0, 1
          LITI    0, 4
          MULI    0, 0
          OFFSET  0, 0
          LITI    0, 8
          STX     0, 0
          POP     0, 1
          LDA     1, 24
          LITI    0, 2
          LITI    0, 4
          MULI    0, 0
          OFFSET  0, 0
          LITI    0, 5
          STX     0, 0
          POP     0, 1
          LDA     1, 24
          LITI    0, 3
          LITI    0, 4
          MULI    0, 0
          OFFSET  0, 0
          LITI    0, 8
          STX     0, 0
          POP     0, 1
          LDA     1, 24
          LITI    0, 4
          LITI    0, 4
          MULI    0, 0
          OFFSET  0, 0
          LITI    0, 2
          STX     0, 0
          POP     0, 1
          LDA     1, 24
          LITI    0, 5
          LITI    0, 4
          MULI    0, 0
          OFFSET  0, 0
          LITI    0, 3
          STX     0, 0
          POP     0, 1
          LDA     1, 24
          LITI    0, 6
          LITI    0, 4
          MULI    0, 0
          OFFSET  0, 0
          LITI    0, 3
          STX     0, 0
          POP     0, 1
          LDA     1, 24
          LITI    0, 7
          LITI    0, 4
          MULI    0, 0
          OFFSET  0, 0
          LITI    0, 9
          STX     0, 0
          POP     0, 1
          LDA     1, 24
          LITI    0, 8
          LITI    0, 4
          MULI    0, 0
          OFFSET  0, 0
          LITI    0, 1
          STX     0, 0
          POP     0, 1
          LDA     1, 24
          LITI    0, 9
          LITI    0, 4
          MULI    0, 0
          OFFSET  0, 0
          LITI    0, 3
          STX     0, 0
```



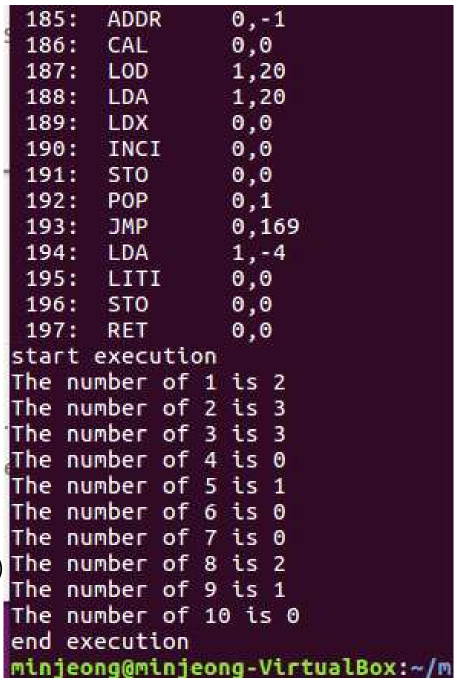
```
                POP     0, 1
                LDA     1, 12
                LITI    0, 0
                STX     0, 0
                POP     0, 1
L2:
                LOD     1, 12
                LITI    0, 10
                LSSI    0, 0
                JPC     0, L3
                LDA     1, 64
                LOD     1, 12
                LITI    0, 4
                MULI    0, 0
                OFFSET  0, 0
                LITI    0, 0
                STX     0, 0
                POP     0, 1
L1:
                LOD     1, 12
                LDA     1, 12
                LDX     0, 0
                INCI    0, 0
                STO     0, 0
                POP     0, 1
                JMP     0, L2
L3:
                LDA     1, 12
                LITI    0, 0
                STX     0, 0
                POP     0, 1
L5:
                LOD     1, 12
                LITI    0, 10
                LSSI    0, 0
                JPC     0, L6
                LDA     1, 16
                LITI    0, 0
                STX     0, 0
                POP     0, 1
L8:
                LOD     1, 16
                LITI    0, 40
                LITI    0, 4
                DIVI    0, 0
                LEQI    0, 0
                JPC     0, L9
                LOD     1, 12
                LITI    0, 1
                ADDI    0, 0
                LDA     1, 24
                LOD     1, 16
                LITI    0, 4
                MULI    0, 0
                OFFSET  0, 0
                LDI     0, 0
                EQLI    0, 0
                JPC     0, L10
                LDA     1, 64
                LOD     1, 12
                LITI    0, 4
                MULI    0, 0
                OFFSET  0, 0
                LDI     0, 0
                LDA     1, 64
                LOD     1, 12
                LITI    0, 4
                MULI    0, 0
                OFFSET  0, 0
                LDX     0, 0
                INCI    0, 0
                STO     0, 0
                POP     0, 1
L10:
L7:
                LOD     1, 16
                LDA     1, 16
                LDX     0, 0
                INCI    0, 0
                STO     0, 0
                POP     0, 1
                JMP     0, L8
L9:
L4:
                LOD     1, 12
                LDA     1, 12
                LDX     0, 0
                INCI    0, 0
                STO     0, 0
                POP     0, 1
                JMP     0, L5
L6:
                LDA     1, 20
                LITI    0, 0
                STX     0, 0
                POP     0, 1
L12:
                LOD     1, 20
                LITI    0, 10
                LSSI    0, 0
                JPC     0, L13
                INT     0, 12
                LDA     0, 12
                LOD     1, 20
                LITI    0, 1
                ADDI    0, 0
                LDA     1, 64
                LOD     1, 20
                LITI    0, 4
                MULI    0, 0
                OFFSET  0, 0
                LDI     0, 0
                POP     0, 6
                ADDR    0, printf
                CAL     0, 0
L11:
                LOD     1, 20
                LDA     1, 20
                LDX     0, 0
                INCI    0, 0
                STO     0, 0
                POP     0, 1
                JMP     0, L12
L13:
                LDA     1, -4
                LITI    0, 0
                STO     0, 0
                RET     0, 0
.literal     12  "The number of %d is %d\n"
~
```

● 어셈블러 프로그램을 시뮬레이터를 이용하여 실행하면 다음과 같은 결과가 나온다.



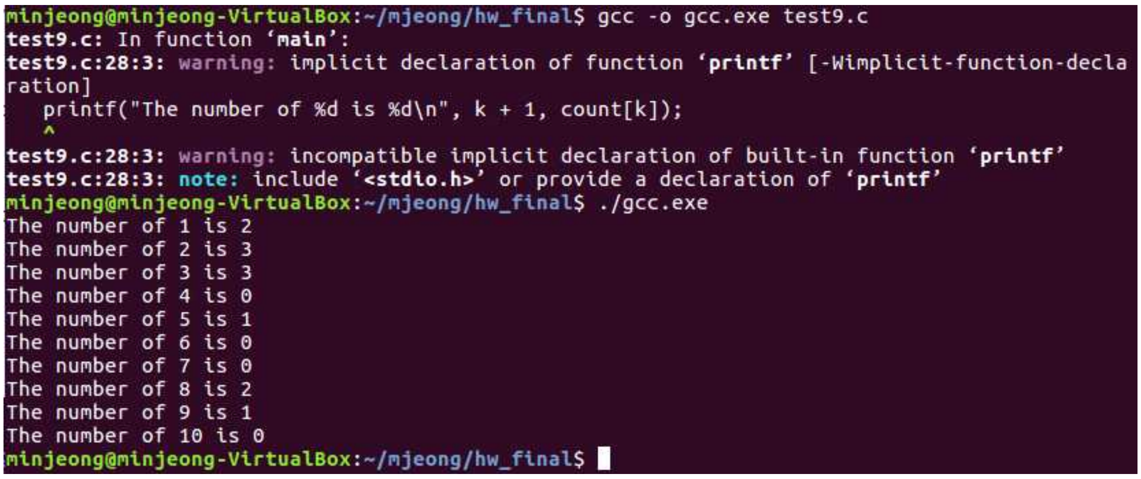
```
======== symbol ========
   1:  printf    -1
   2:  malloc    -2
   3:  scanf     -3
   4:  main      3
   5:  L2        88
   6:  L3        107
   7:  L1        100
   8:  L5        111
   9:  L6        165
  10:  L8        119
  11:  L9        158
  12:  L10       151
  13:  L7        151
  14:  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
```

(생략)



```
 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
```

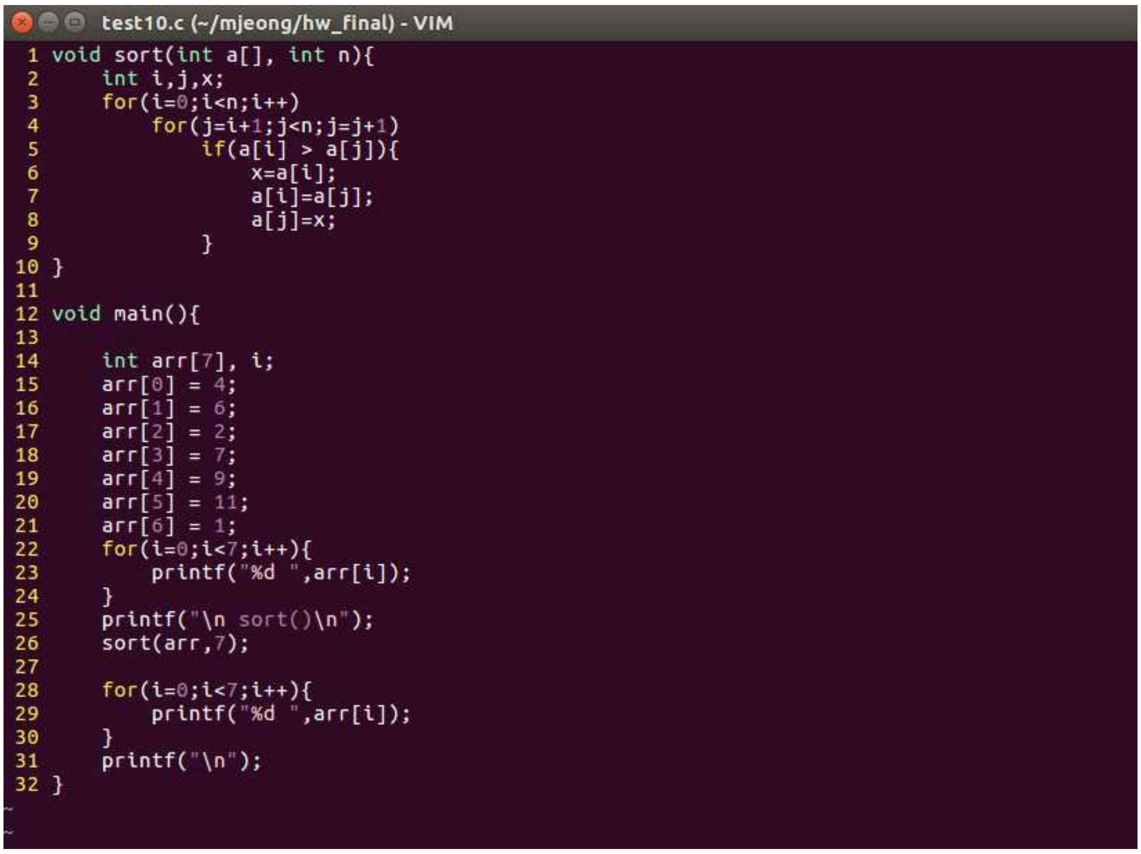
test9.c를 gcc로 실행하면 다음과 같은 결과가 나온다. 두 개가 동일한 결과라는 것을 확인할 수 있다.



```
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$ gcc -o gcc.exe test9.c
test9.c: In function ‘main’:
test9.c:28:3: warning: implicit declaration of function ‘printf’ [-Wimplicit-function-decla
ration]
   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:~/mjeong/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:~/mjeong/hw_final$ 
```

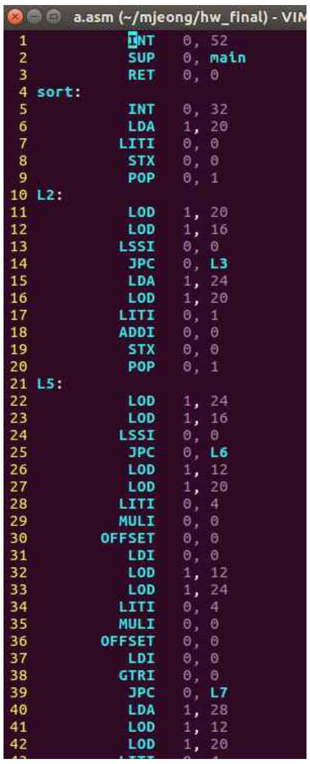
10) test10.c

배열안의 값을 오름차순으로 정렬하는 프로그램이다.



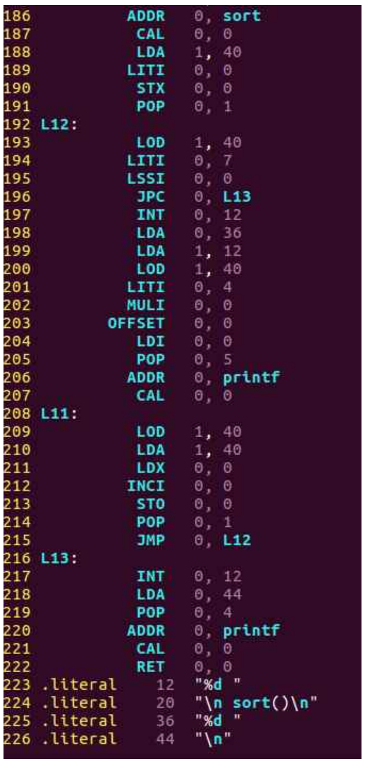
```c
void sort(int a[], int n){
    int i,j,x;
    for(i=0;i<n;i++)
        for(j=i+1;j<n;j=j+1)
            if(a[i] > a[j]){
                x=a[i];
                a[i]=a[j];
                a[j]=x;
            }
}

void main(){

    int arr[7], i;
    arr[0] = 4;
    arr[1] = 6;
    arr[2] = 2;
    arr[3] = 7;
    arr[4] = 9;
    arr[5] = 11;
    arr[6] = 1;
    for(i=0;i<7;i++){
        printf("%d ",arr[i]);
    }
    printf("\n sort()\n");
    sort(arr,7);

    for(i=0;i<7;i++){
        printf("%d ",arr[i]);
    }
    printf("\n");
}
~
~
```

● 컴파일러를 통하여 만들어진 어셈블러 프로그램은 다음과 같다.



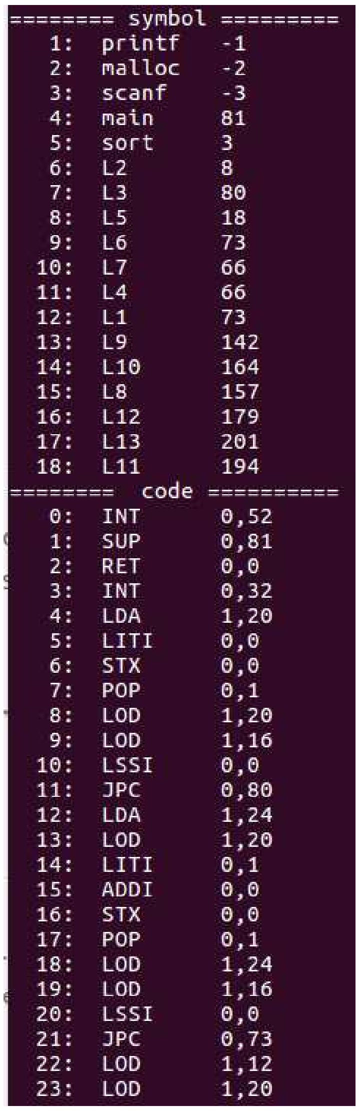
```
            INT    0, 52
            SUP    0, main
            RET    0, 0
sort:
            INT    0, 32
            LDA    1, 20
           LITI    0, 0
            STX    0, 0
            POP    0, 1
L2:
            LOD    1, 20
            LOD    1, 16
           LSSI    0, 0
            JPC    0, L3
            LDA    1, 24
            LOD    1, 20
           LITI    0, 1
           ADDI    0, 0
            STX    0, 0
            POP    0, 1
L5:
            LOD    1, 24
            LOD    1, 16
           LSSI    0, 0
            JPC    0, L6
            LOD    1, 12
            LOD    1, 20
           LITI    0, 4
           MULI    0, 0
         OFFSET    0, 0
            LDI    0, 0
            LOD    1, 12
            LOD    1, 24
           LITI    0, 4
           MULI    0, 0
         OFFSET    0, 0
            LDI    0, 0
           GTRI    0, 0
            JPC    0, L7
            LDA    1, 28
            LOD    1, 12
            LOD    1, 20
```

(생략)



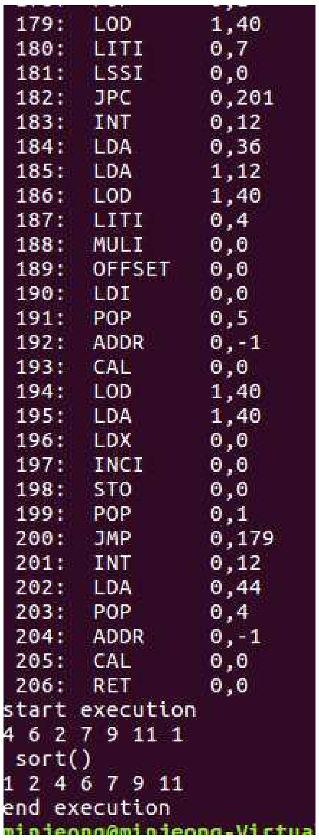
```
            ADDR   0, sort
             CAL   0, 0
             LDA   1, 40
            LITI   0, 0
             STX   0, 0
             POP   0, 1
L12:
             LOD   1, 40
            LITI   0, 7
            LSSI   0, 0
             JPC   0, L13
             INT   0, 12
             LDA   0, 36
             LDA   1, 12
             LOD   1, 40
            LITI   0, 4
            MULI   0, 0
          OFFSET   0, 0
             LDI   0, 0
             POP   0, 5
            ADDR   0, printf
             CAL   0, 0
L11:
             LOD   1, 40
             LDA   1, 40
             LDX   0, 0
            INCI   0, 0
             STO   0, 0
             POP   0, 1
             JMP   0, L12
L13:
             INT   0, 12
             LDA   0, 44
             POP   0, 4
            ADDR   0, printf
             CAL   0, 0
             RET   0, 0
.literal      12   "%d "
.literal      20   "\n sort()\n"
.literal      36   "%d "
.literal      44   "\n"
```

● 어셈블러 프로그램을 시뮬레이터를 이용하여 실행하면 다음과 같은 결과가 나온다.



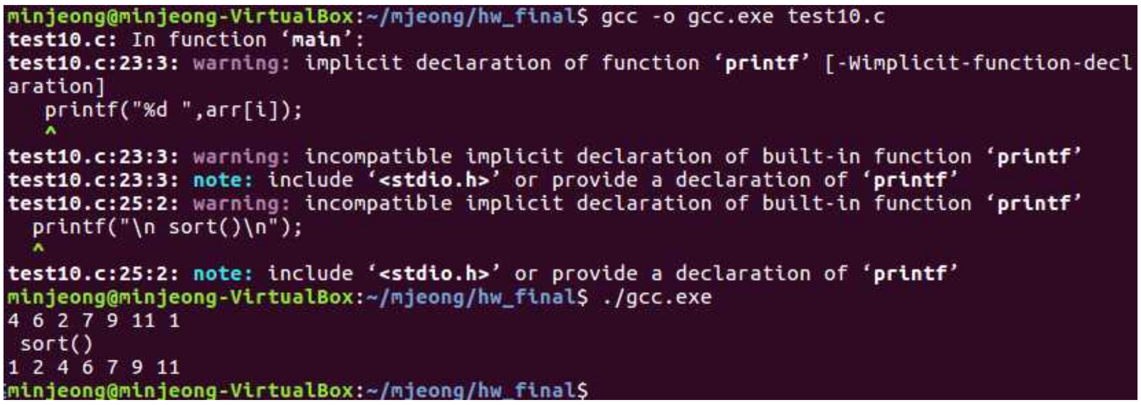
```
======== symbol =========
   1:  printf    -1
   2:  malloc    -2
   3:  scanf     -3
   4:  main      81
   5:  sort      3
   6:  L2        8
   7:  L3        80
   8:  L5        18
   9:  L6        73
  10:  L7        66
  11:  L4        66
  12:  L1        73
  13:  L9        142
  14:  L10       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
```

(생략)



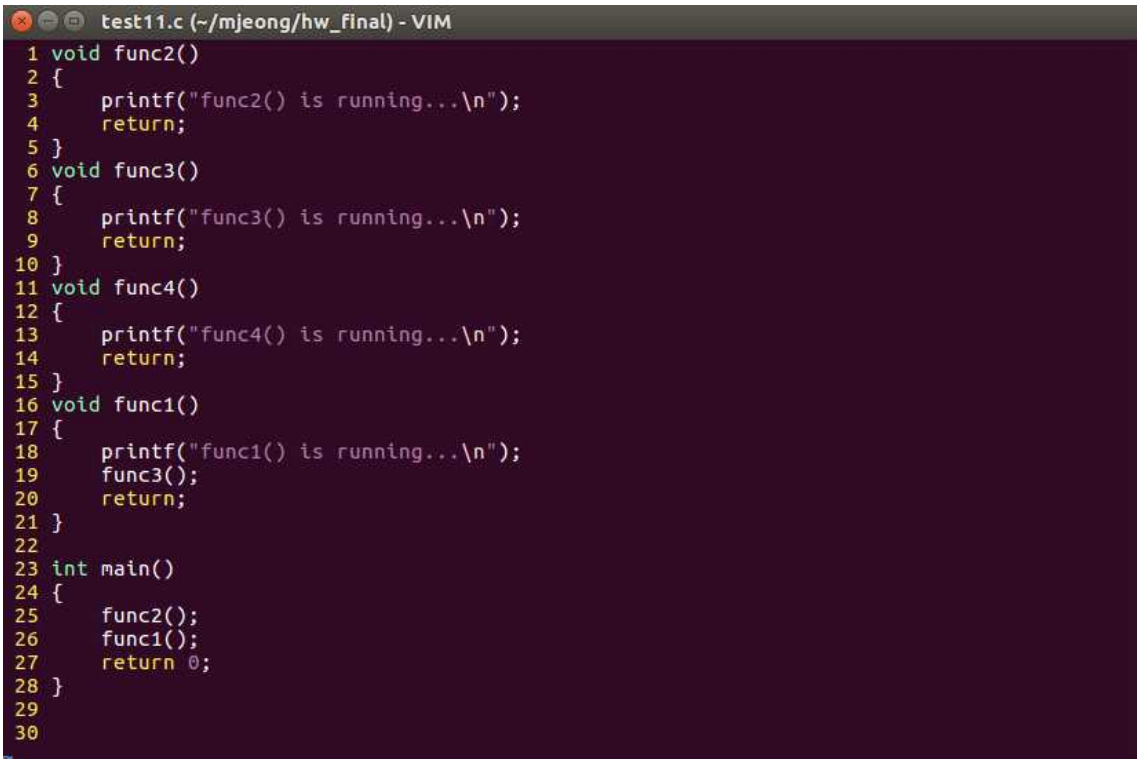
```
 179:  LOD       1,40
 180:  LITI      0,7
 181:  LSSI      0,0
 182:  JPC       0,201
 183:  INT       0,12
 184:  LDA       0,36
 185:  LDA       1,12
 186:  LOD       1,40
 187:  LITI      0,4
 188:  MULI      0,0
 189:  OFFSET    0,0
 190:  LDI       0,0
 191:  POP       0,5
 192:  ADDR      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
```

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$
```

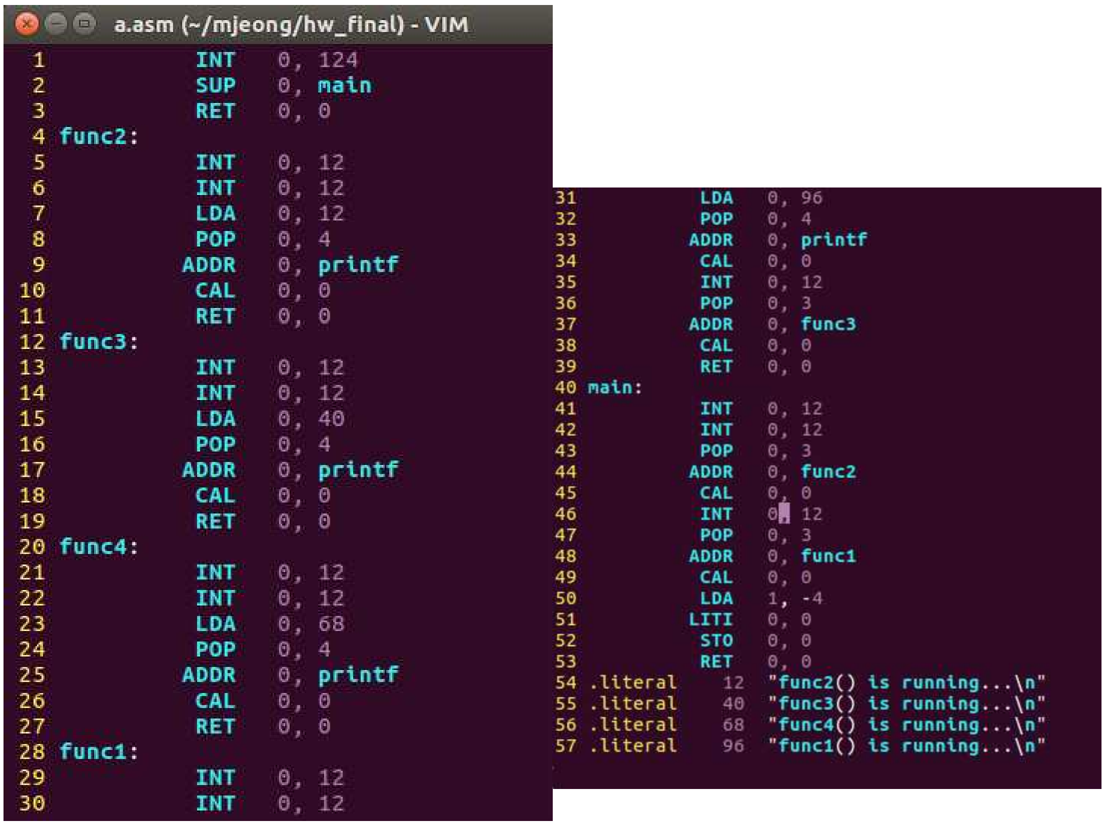
11) test11.c



test11.c (~/mjeong/hw_final) - VIM

```
void func2()
{
	printf("func2() is running...\n");
	return;
}
void func3()
{
	printf("func3() is running...\n");
	return;
}
void func4()
{
	printf("func4() is running...\n");
	return;
}
void func1()
{
	printf("func1() is running...\n");
	func3();
	return;
}

int main()
{
	func2();
	func1();
	return 0;
}


```

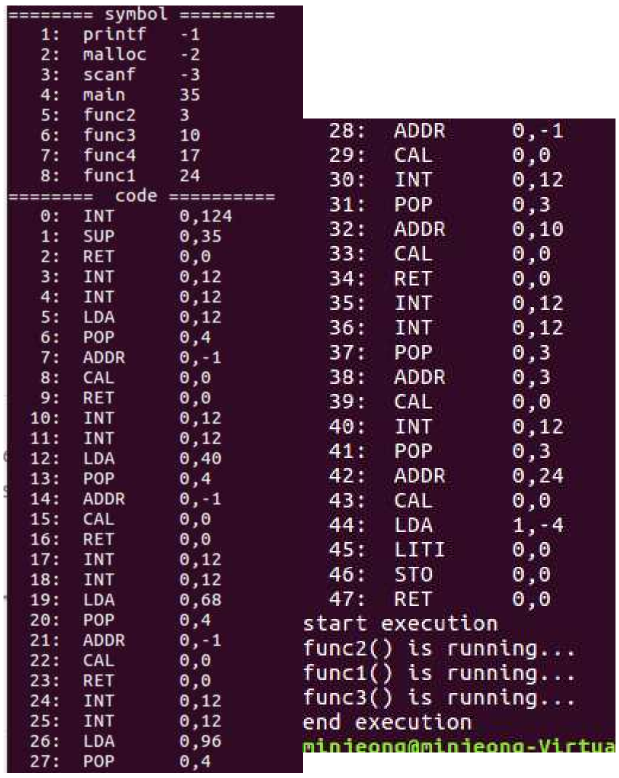
● 컴파일러를 통하여 만들어진 어셈블러 프로그램은 다음과 같다.



a.asm (~/mjeong/hw_final) - VIM

```
		INT	0, 124
		SUP	0, main
		RET	0, 0
func2:
		INT	0, 12
		INT	0, 12
		LDA	0, 12
		POP	0, 4
		ADDR	0, printf
		CAL	0, 0
		RET	0, 0
func3:
		INT	0, 12
		INT	0, 12
		LDA	0, 40
		POP	0, 4
		ADDR	0, printf
		CAL	0, 0
		RET	0, 0
func4:
		INT	0, 12
		INT	0, 12
		LDA	0, 68
		POP	0, 4
		ADDR	0, printf
		CAL	0, 0
		RET	0, 0
func1:
		INT	0, 12
		INT	0, 12
		LDA	0, 96
		POP	0, 4
		ADDR	0, printf
		CAL	0, 0
		INT	0, 12
		POP	0, 3
		ADDR	0, func3
		CAL	0, 0
		RET	0, 0
main:
		INT	0, 12
		INT	0, 12
		POP	0, 3
		ADDR	0, func2
		CAL	0, 0
		INT	0, 12
		POP	0, 3
		ADDR	0, func1
		CAL	0, 0
		LDA	1, -4
		LITI	0, 0
		STO	0, 0
		RET	0, 0
.literal	12	"func2() is running...\n"
.literal	40	"func3() is running...\n"
.literal	68	"func4() is running...\n"
.literal	96	"func1() is running...\n"
```

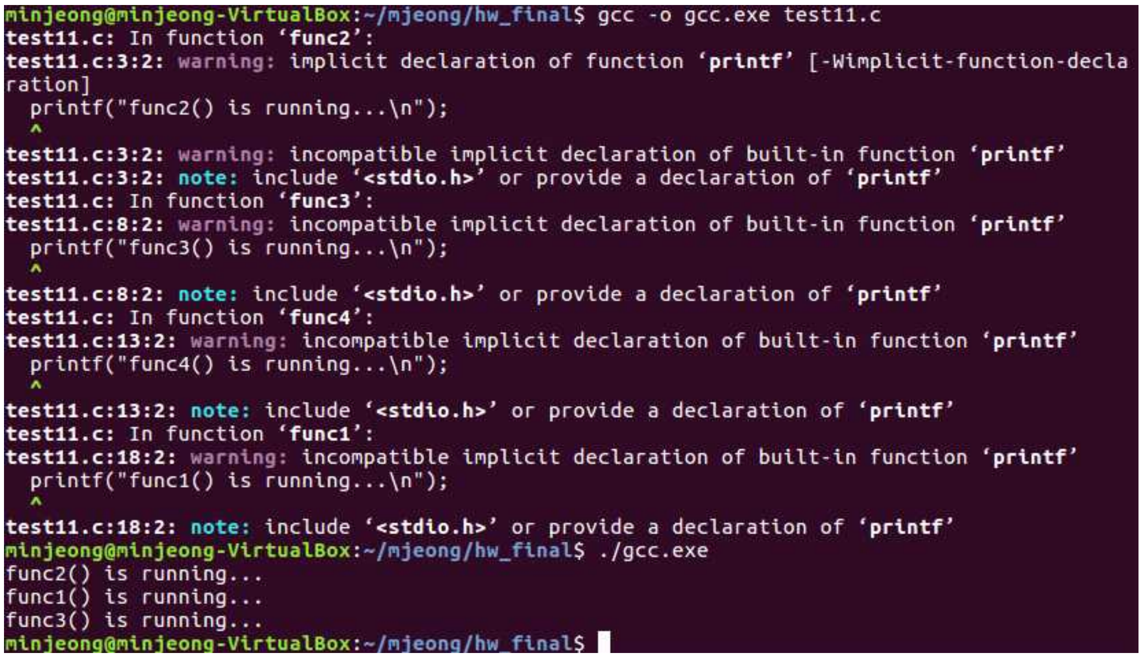
● 어셈블러 프로그램을 시뮬레이터를 이용하여 실행하면 다음과 같은 결과가 나온다.



```
======== 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
```

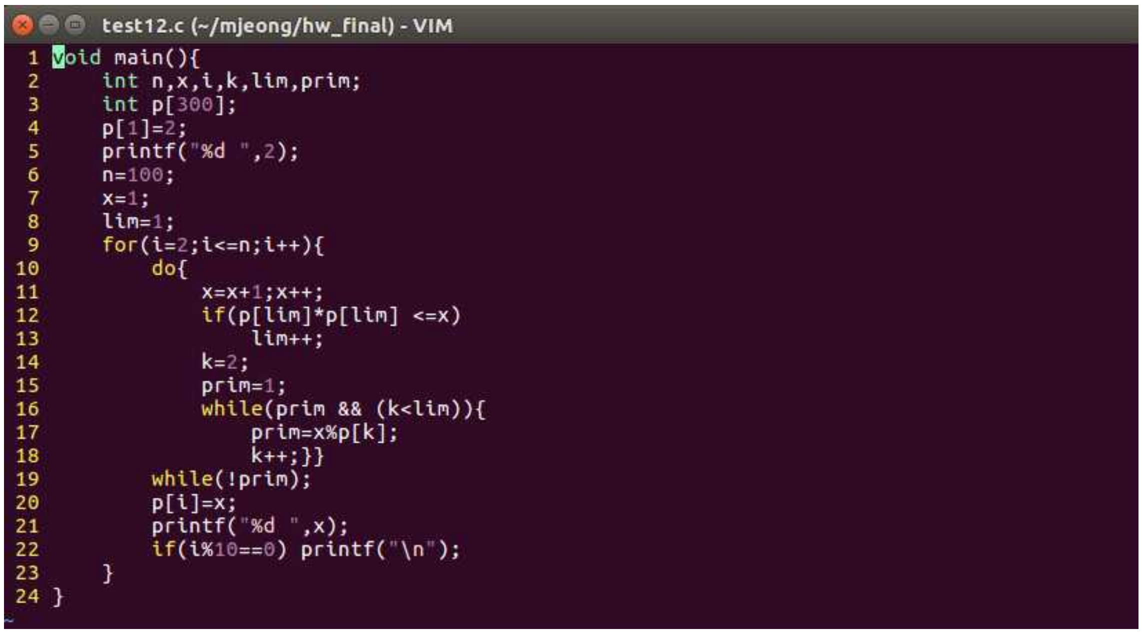
test11.c를 gcc로 실행하면 다음과 같은 결과가 나온다. 두 개가 동일한 결과라는 것을 확인할 수 있다.



```
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-decla
ration]
  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$
```

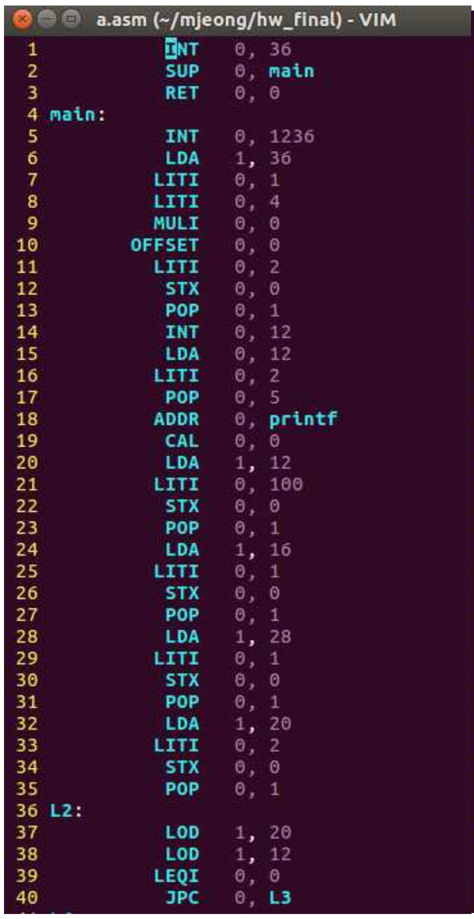
12) test12.c

소수를 구하는 프로그램이다.

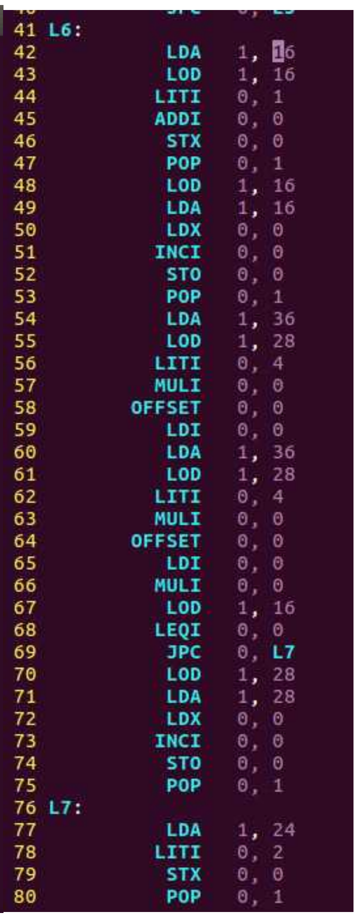

test12.c (~/mjeong/hw_final) - VIM

```c
void main(){
    int n,x,i,k,lim,prim;
    int p[300];
    p[1]=2;
    printf("%d ",2);
    n=100;
    x=1;
    lim=1;
    for(i=2;i<=n;i++){
        do{
            x=x+1;x++;
            if(p[lim]*p[lim] <=x)
                lim++;
            k=2;
            prim=1;
            while(prim && (k<lim)){
                prim=x%p[k];
                k++;}}
        while(!prim);
        p[i]=x;
        printf("%d ",x);
        if(i%10==0) printf("\n");
    }
}
```

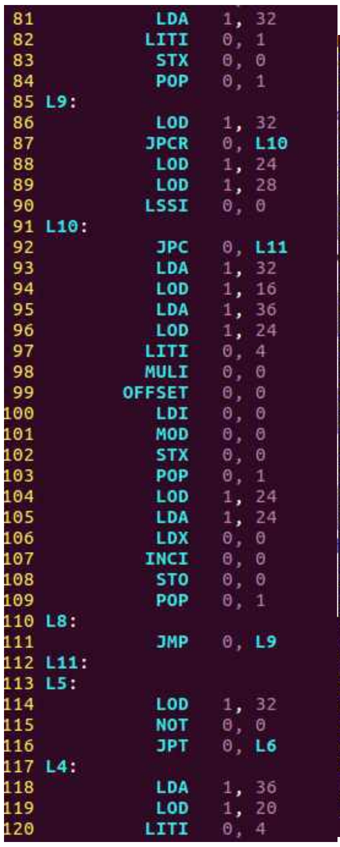
● 컴파일러를 통하여 만들어진 어셈블러 프로그램은 다음과 같다.


a.asm (~/mjeong/hw_final) - VIM

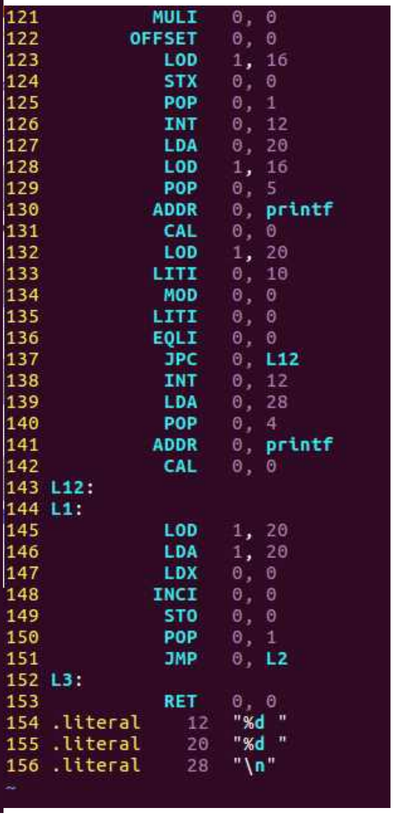
```
            INT     0, 36
            SUP     0, main
            RET     0, 0
main:
            INT     0, 1236
            LDA     1, 36
            LITI    0, 1
            LITI    0, 4
            MULI    0, 0
            OFFSET  0, 0
            LITI    0, 2
            STX     0, 0
            POP     0, 1
            INT     0, 12
            LDA     0, 12
            LITI    0, 2
            POP     0, 5
            ADDR    0, printf
            CAL     0, 0
            LDA     1, 12
            LITI    0, 100
            STX     0, 0
            POP     0, 1
            LDA     1, 16
            LITI    0, 1
            STX     0, 0
            POP     0, 1
            LDA     1, 28
            LITI    0, 1
            STX     0, 0
            POP     0, 1
            LDA     1, 20
            LITI    0, 2
            STX     0, 0
            POP     0, 1
L2:
            LOD     1, 20
            LOD     1, 12
            LEQI    0, 0
            JPC     0, L3
```



```
L6:
            LDA     1, 16
            LOD     1, 16
            LITI    0, 1
            ADDI    0, 0
            STX     0, 0
            POP     0, 1
            LOD     1, 16
            LDA     1, 16
            LDX     0, 0
            INCI    0, 0
            STO     0, 0
            POP     0, 1
            LDA     1, 36
            LOD     1, 28
            LITI    0, 4
            MULI    0, 0
            OFFSET  0, 0
            LDI     0, 0
            LDA     1, 36
            LOD     1, 28
            LITI    0, 4
            MULI    0, 0
            OFFSET  0, 0
            LDI     0, 0
            MULI    0, 0
            LOD     1, 16
            LEQI    0, 0
            JPC     0, L7
            LOD     1, 28
            LDA     1, 28
            LDX     0, 0
            INCI    0, 0
            STO     0, 0
            POP     0, 1
L7:
            LDA     1, 24
            LITI    0, 2
            STX     0, 0
            POP     0, 1
```

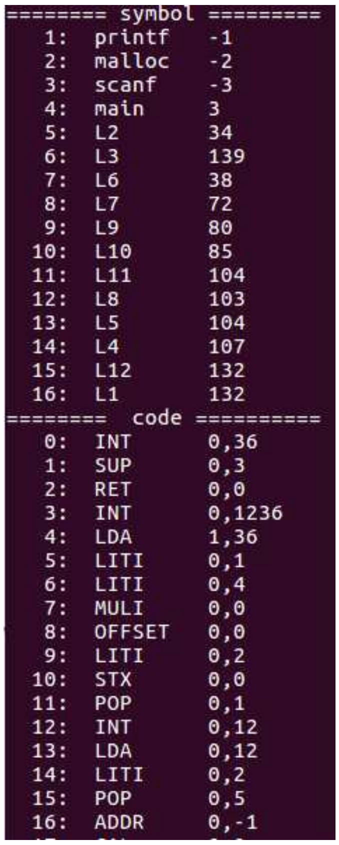


```
        LDA     1, 32
        LITI    0, 1
        STX     0, 0
        POP     0, 1
L9:
        LOD     1, 32
        JPCR    0, L10
        LOD     1, 24
        LOD     1, 28
        LSSI    0, 0
L10:
        JPC     0, L11
        LDA     1, 32
        LOD     1, 16
        LDA     1, 36
        LOD     1, 24
        LITI    0, 4
        MULI    0, 0
      OFFSET    0, 0
        LDI     0, 0
        MOD     0, 0
        STX     0, 0
        POP     0, 1
        LOD     1, 24
        LDA     1, 24
        LDX     0, 0
        INCI    0, 0
        STO     0, 0
        POP     0, 1
L8:
        JMP     0, L9
L11:
L5:
        LOD     1, 32
        NOT     0, 0
        JPT     0, L6
L4:
        LDA     1, 36
        LOD     1, 20
        LITI    0, 4
```



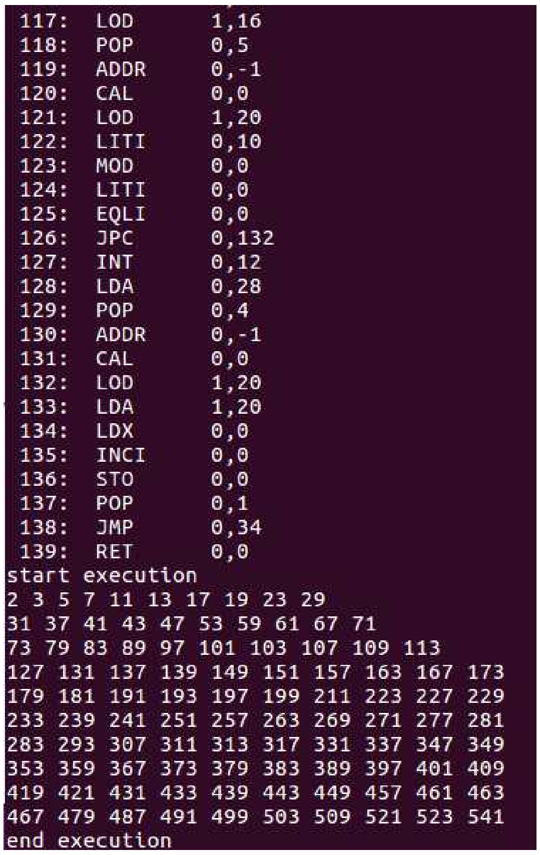
```
        MULI    0, 0
      OFFSET    0, 0
        LOD     1, 16
        STX     0, 0
        POP     0, 1
        INT     0, 12
        LDA     0, 20
        LOD     1, 16
        POP     0, 5
        ADDR    0, printf
        CAL     0, 0
        LOD     1, 20
        LITI    0, 10
        MOD     0, 0
        LITI    0, 0
        EQLI    0, 0
        JPC     0, L12
        INT     0, 12
        LDA     0, 28
        POP     0, 4
        ADDR    0, printf
        CAL     0, 0
L12:
L1:
        LOD     1, 20
        LDA     1, 20
        LDX     0, 0
        INCI    0, 0
        STO     0, 0
        POP     0, 1
        JMP     0, L2
L3:
        RET     0, 0
.literal    12  "%d "
.literal    20  "%d "
.literal    28  "\n"
~
```

● 어셈블러 프로그램을 시뮬레이터를 이용하여 실행하면 다음과 같은 결과가 나온다.



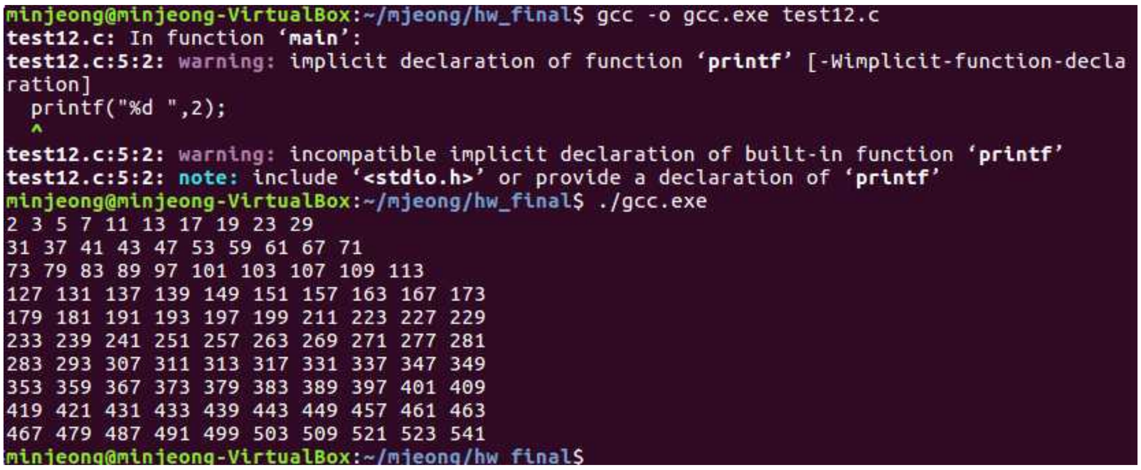
```
======== symbol =========
   1:  printf    -1
   2:  malloc    -2
   3:  scanf     -3
   4:  main      3
   5:  L2        34
   6:  L3        139
   7:  L6        38
   8:  L7        72
   9:  L9        80
  10:  L10       85
  11:  L11       104
  12:  L8        103
  13:  L5        104
  14:  L4        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)



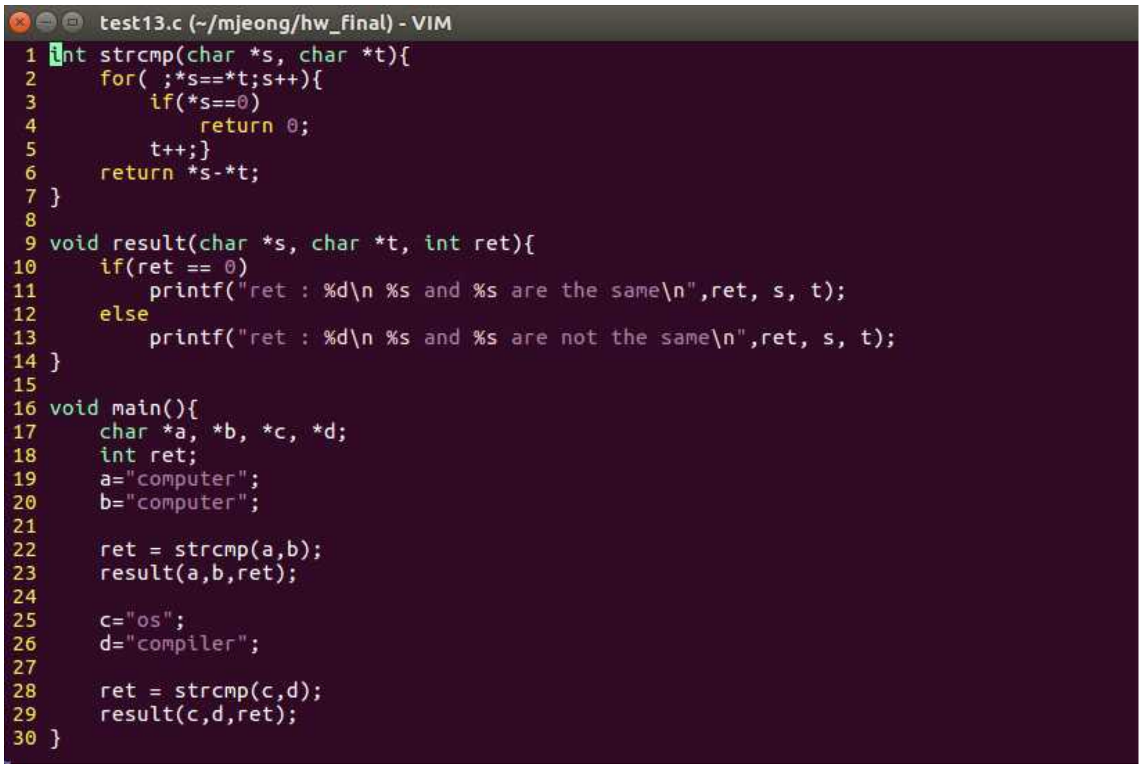
```
 117:  LOD       1,16
 118:  POP       0,5
 119:  ADDR      0,-1
 120:  CAL       0,0
 121:  LOD       1,20
 122:  LITI      0,10
 123:  MOD       0,0
 124:  LITI      0,0
 125:  EQLI      0,0
 126:  JPC       0,132
 127:  INT       0,12
 128:  LDA       0,28
 129:  POP       0,4
 130:  ADDR      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.



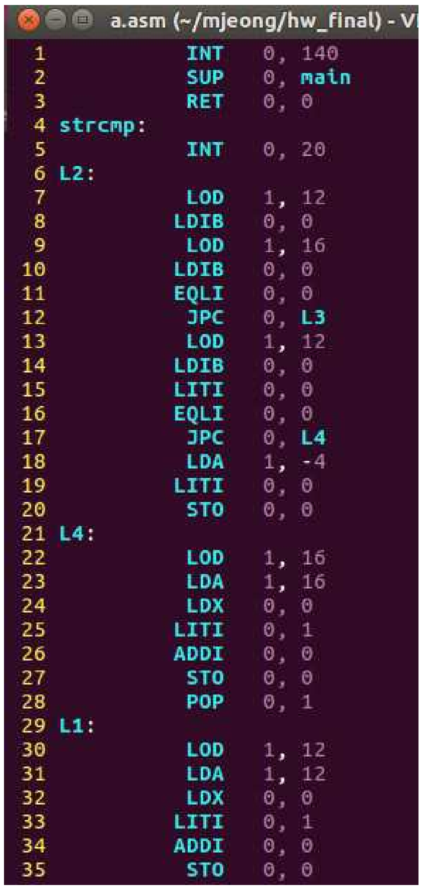
```
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$ gcc -o gcc.exe test12.c
test12.c: In function ‘main’:
test12.c:5:2: warning: implicit declaration of function ‘printf’ [-Wimplicit-function-decla
ration]
  printf("%d ",2);
  ^
test12.c:5:2: warning: incompatible implicit declaration of built-in function ‘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

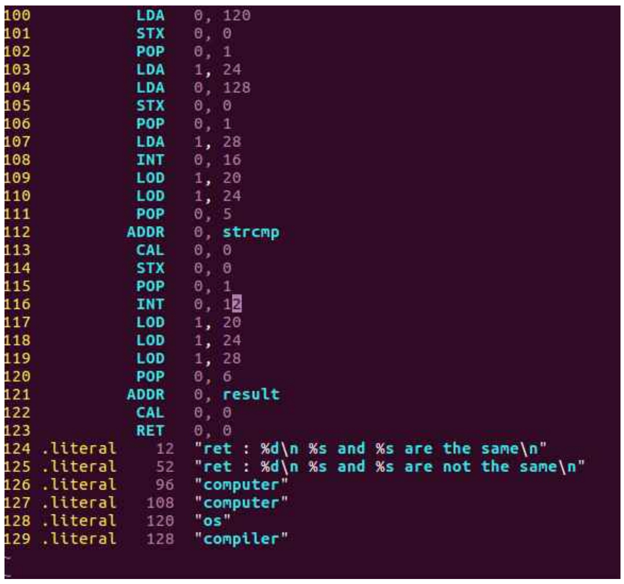
```c
int strcmp(char *s, char *t){
    for( ;*s==*t;s++){
        if(*s==0)
            return 0;
        t++;}
    return *s-*t;
}

void result(char *s, char *t, int ret){
    if(ret == 0)
        printf("ret : %d\n %s and %s are the same\n",ret, s, t);
    else
        printf("ret : %d\n %s and %s are not the same\n",ret, s, t);
}

void main(){
    char *a, *b, *c, *d;
    int ret;
    a="computer";
    b="computer";

    ret = strcmp(a,b);
    result(a,b,ret);

    c="os";
    d="compiler";

    ret = strcmp(c,d);
    result(c,d,ret);
}
```

● 컴파일러를 통하여 만들어진 어셈블러 프로그램은 다음과 같다.


a.asm (~/mjeong/hw_final) - Vi

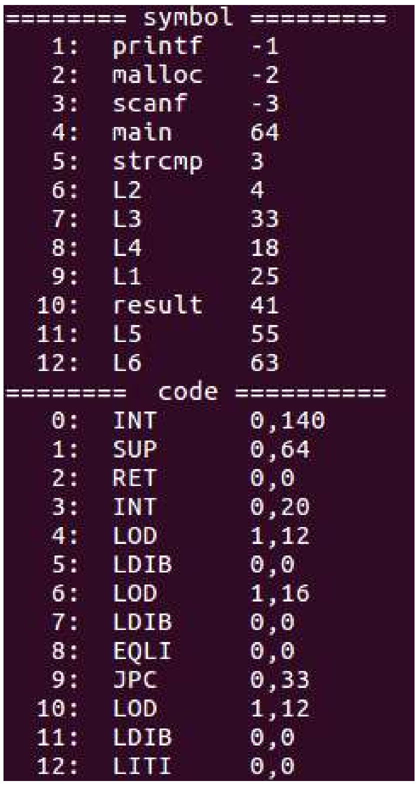
```
             INT    0, 140
             SUP    0, main
             RET    0, 0
strcmp:
             INT    0, 20
L2:
             LOD    1, 12
            LDIB    0, 0
             LOD    1, 16
            LDIB    0, 0
            EQLI    0, 0
             JPC    0, L3
             LOD    1, 12
            LDIB    0, 0
            LITI    0, 0
            EQLI    0, 0
             JPC    0, L4
             LDA    1, -4
            LITI    0, 0
             STO    0, 0
L4:
             LOD    1, 16
             LDA    1, 16
             LDX    0, 0
            LITI    0, 1
            ADDI    0, 0
             STO    0, 0
             POP    0, 1
L1:
             LOD    1, 12
             LDA    1, 12
             LDX    0, 0
            LITI    0, 1
            ADDI    0, 0
             STO    0, 0
```

(생략)



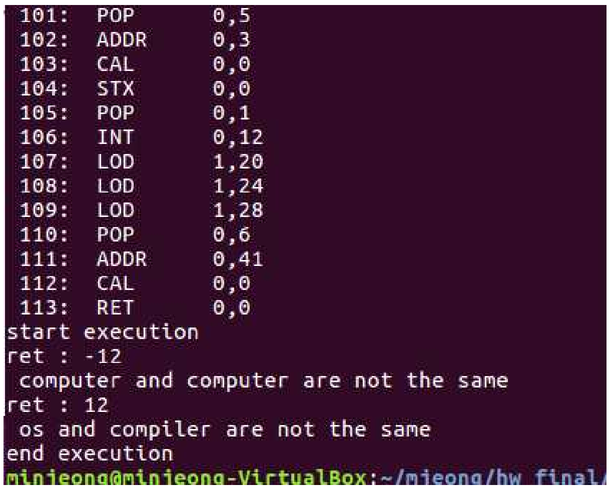
```
             LDA    0, 120
             STX    0, 0
             POP    0, 1
             LDA    1, 24
             LDA    0, 128
             STX    0, 0
             POP    0, 1
             LDA    1, 28
             INT    0, 16
             LOD    1, 20
             LOD    1, 24
             POP    0, 5
            ADDR    0, strcmp
             CAL    0, 0
             STX    0, 0
             POP    0, 1
             INT    0, 12
             LOD    1, 20
             LOD    1, 24
             LOD    1, 28
             POP    0, 6
            ADDR    0, result
             CAL    0, 0
             RET    0, 0
 .literal      12   "ret : %d\n %s and %s are the same\n"
 .literal      52   "ret : %d\n %s and %s are not the same\n"
 .literal      96   "computer"
 .literal     108   "computer"
 .literal     120   "os"
 .literal     128   "compiler"
```

● 어셈블러 프로그램을 시뮬레이터를 이용하여 실행하면 다음과 같은 결과가 나온다.



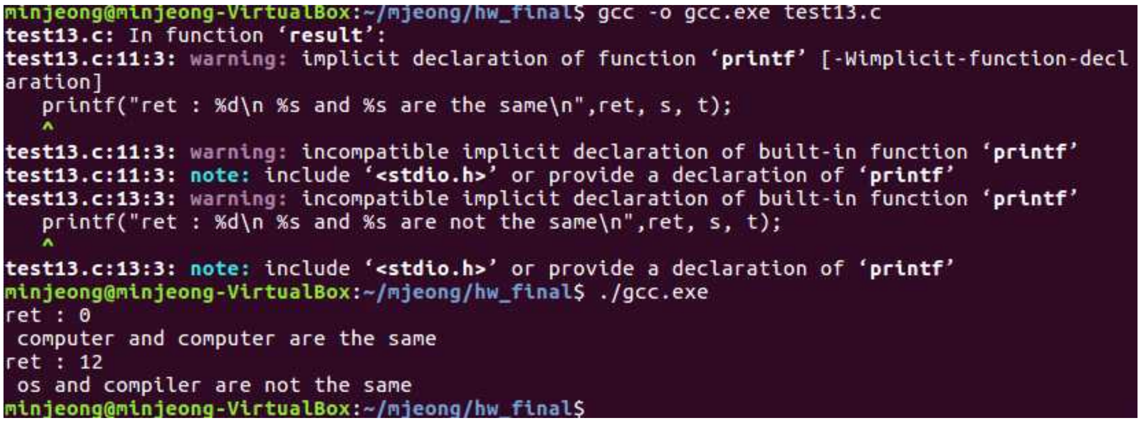
```
======== symbol =========
   1:  printf   -1
   2:  malloc   -2
   3:  scanf    -3
   4:  main     64
   5:  strcmp   3
   6:  L2       4
   7:  L3       33
   8:  L4       18
   9:  L1       25
  10:  result   41
  11:  L5       55
  12:  L6       63
========  code ==========
   0:  INT      0,140
   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)



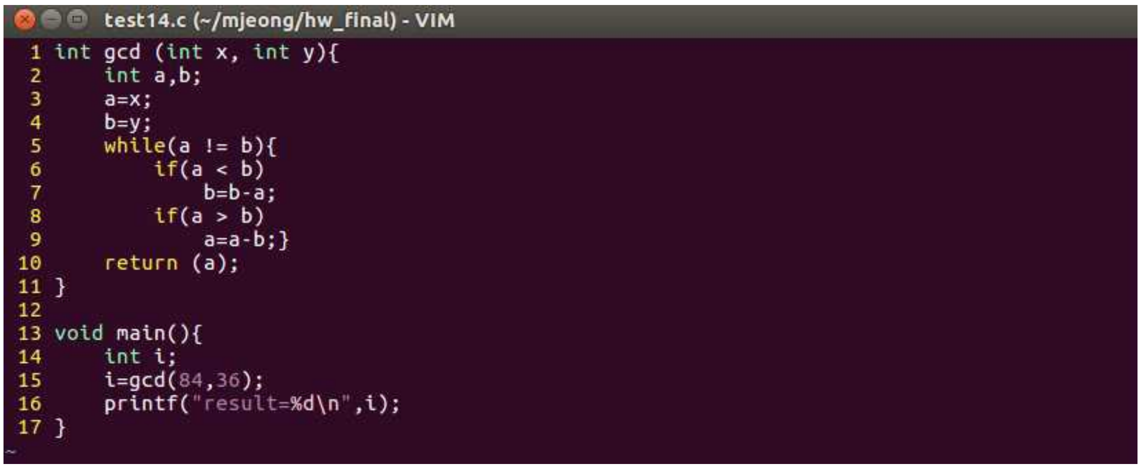
```
 101:  POP      0,5
 102:  ADDR     0,3
 103:  CAL      0,0
 104:  STX      0,0
 105:  POP      0,1
 106:  INT      0,12
 107:  LOD      1,20
 108:  LOD      1,24
 109:  LOD      1,28
 110:  POP      0,6
 111:  ADDR     0,41
 112:  CAL      0,0
 113:  RET      0,0
start execution
ret : -12
 computer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

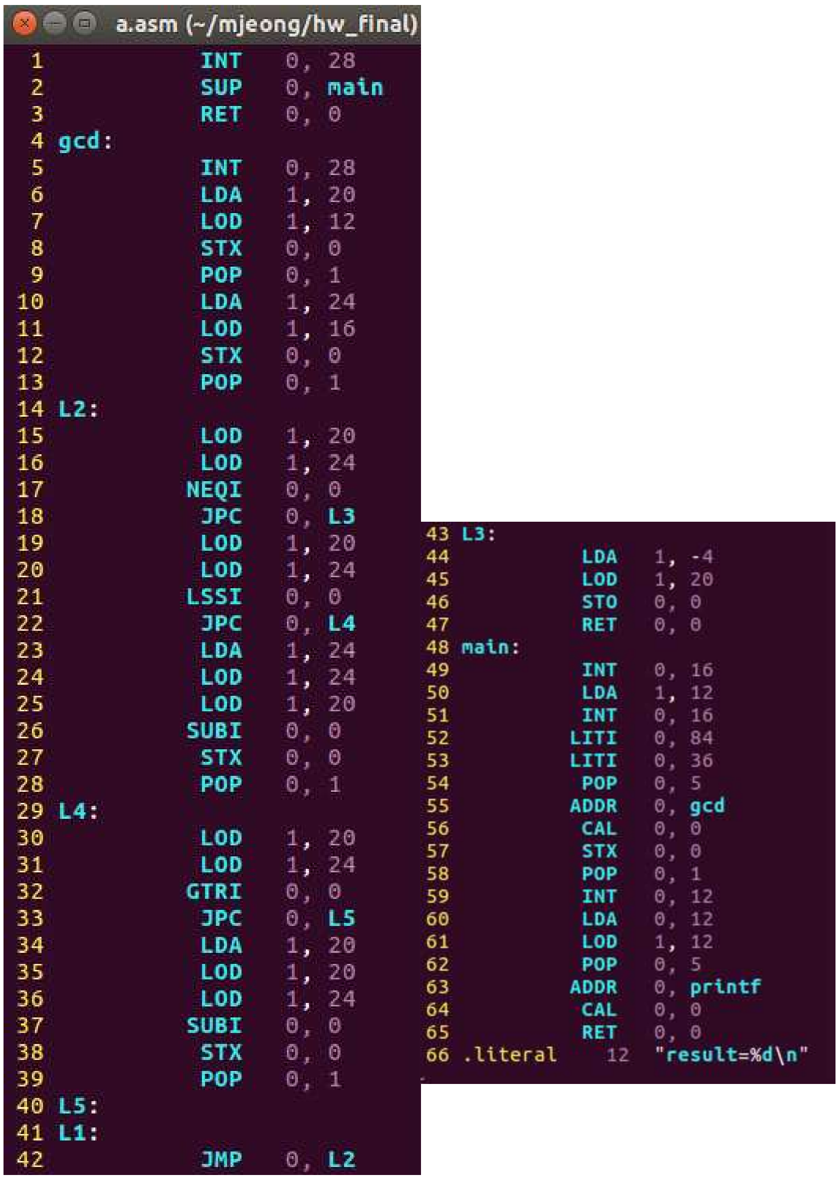
14) test14.c



test14.c (~/mjeong/hw_final) - VIM

```c
int gcd (int x, int y){
    int a,b;
    a=x;
    b=y;
    while(a != b){
        if(a < b)
            b=b-a;
        if(a > b)
            a=a-b;}
    return (a);
}

void main(){
    int i;
    i=gcd(84,36);
    printf("result=%d\n",i);
}
```

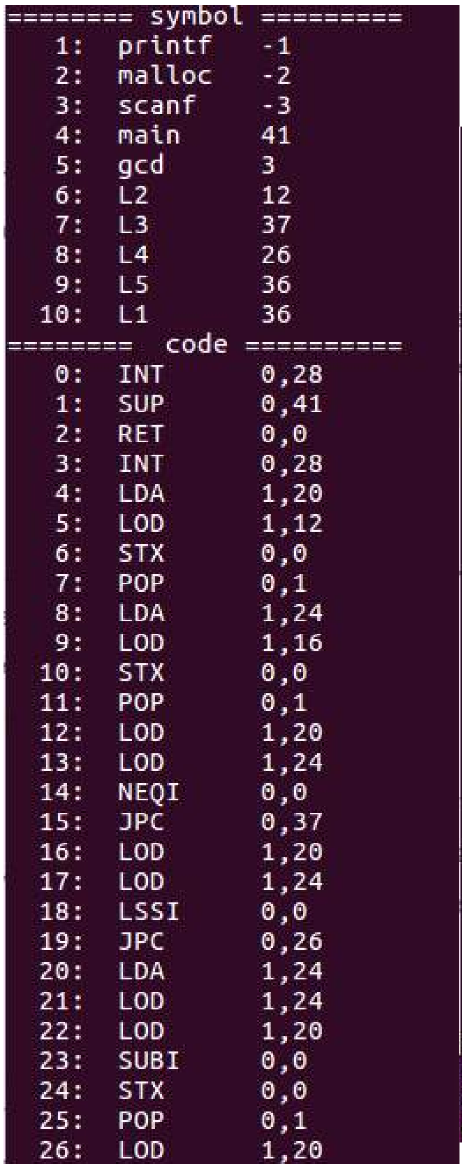
● 컴파일러를 통하여 만들어진 어셈블러 프로그램은 다음과 같다.



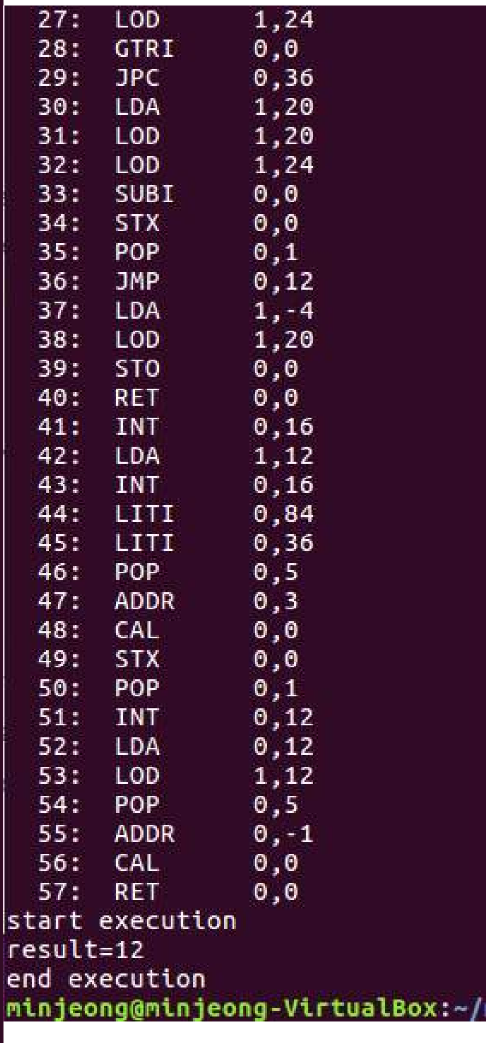
a.asm (~/mjeong/hw_final)

```
            INT     0, 28
            SUP     0, main
            RET     0, 0
gcd:
            INT     0, 28
            LDA     1, 20
            LOD     1, 12
            STX     0, 0
            POP     0, 1
            LDA     1, 24
            LOD     1, 16
            STX     0, 0
            POP     0, 1
L2:
            LOD     1, 20
            LOD     1, 24
            NEQI    0, 0
            JPC     0, L3
            LOD     1, 20
            LOD     1, 24
            LSSI    0, 0
            JPC     0, L4
            LDA     1, 24
            LOD     1, 24
            LOD     1, 20
            SUBI    0, 0
            STX     0, 0
            POP     0, 1
L4:
            LOD     1, 20
            LOD     1, 24
            GTRI    0, 0
            JPC     0, L5
            LDA     1, 20
            LOD     1, 20
            LOD     1, 24
            SUBI    0, 0
            STX     0, 0
            POP     0, 1
L5:
L1:
            JMP     0, L2
L3:
            LDA     1, -4
            LOD     1, 20
            STO     0, 0
            RET     0, 0
main:
            INT     0, 16
            LDA     1, 12
            INT     0, 16
            LITI    0, 84
            LITI    0, 36
            POP     0, 5
            ADDR    0, gcd
            CAL     0, 0
            STX     0, 0
            POP     0, 1
            INT     0, 12
            LDA     0, 12
            LOD     1, 12
            POP     0, 5
            ADDR    0, printf
            CAL     0, 0
            RET     0, 0
.literal        12  "result=%d\n"
```

● 어셈블러 프로그램을 시뮬레이터를 이용하여 실행하면 다음과 같은 결과가 나온다.

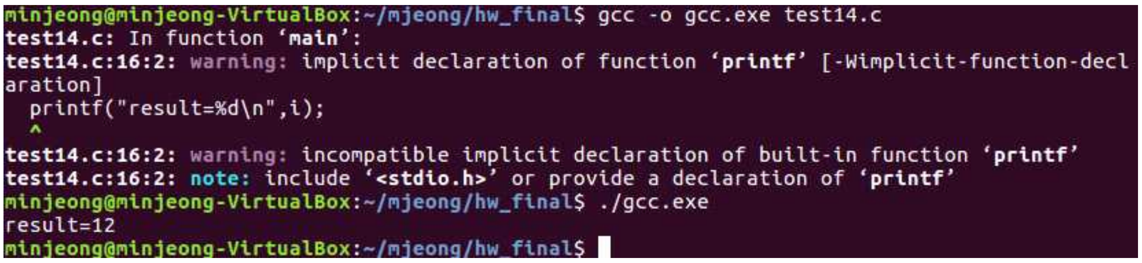


```
======== symbol =========
   1:  printf    -1
   2:  malloc    -2
   3:  scanf     -3
   4:  main      41
   5:  gcd       3
   6:  L2        12
   7:  L3        37
   8:  L4        26
   9:  L5        36
  10:  L1        36
========  code ==========
   0:  INT       0,28
   1:  SUP       0,41
   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
```



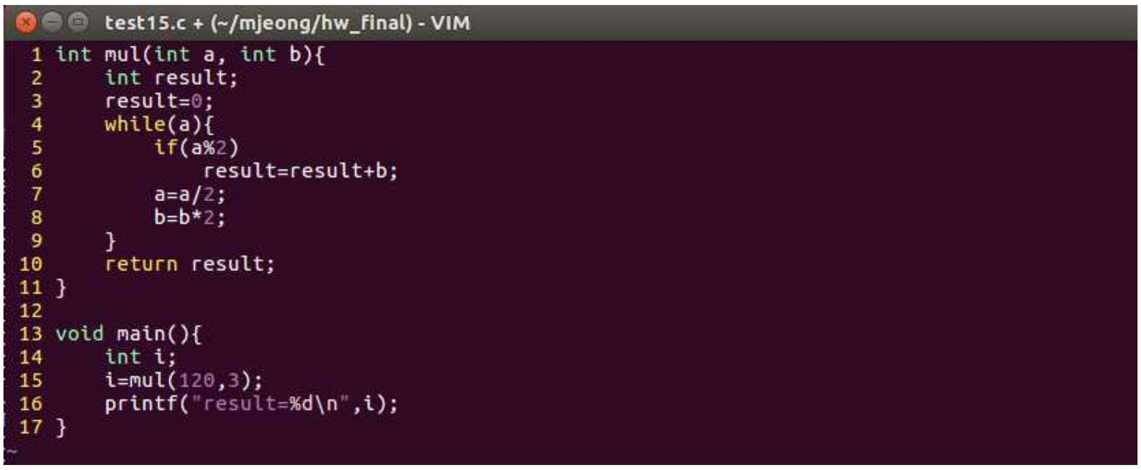
```
  27:  LOD       1,24
  28:  GTRI      0,0
  29:  JPC       0,36
  30:  LDA       1,20
  31:  LOD       1,20
  32:  LOD       1,24
  33:  SUBI      0,0
  34:  STX       0,0
  35:  POP       0,1
  36:  JMP       0,12
  37:  LDA       1,-4
  38:  LOD       1,20
  39:  STO       0,0
  40:  RET       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.



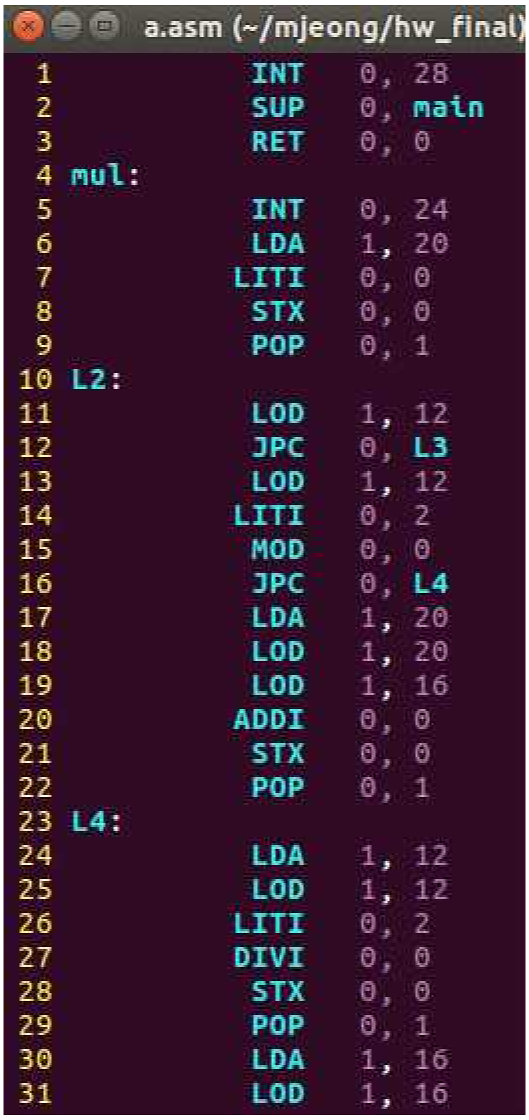
```
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$ gcc -o gcc.exe test14.c
test14.c: In function 'main':
test14.c:16:2: warning: implicit declaration of function 'printf' [-Wimplicit-function-decl
aration]
  printf("result=%d\n",i);
  ^
test14.c:16:2: warning: incompatible implicit declaration of built-in 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$ 
```

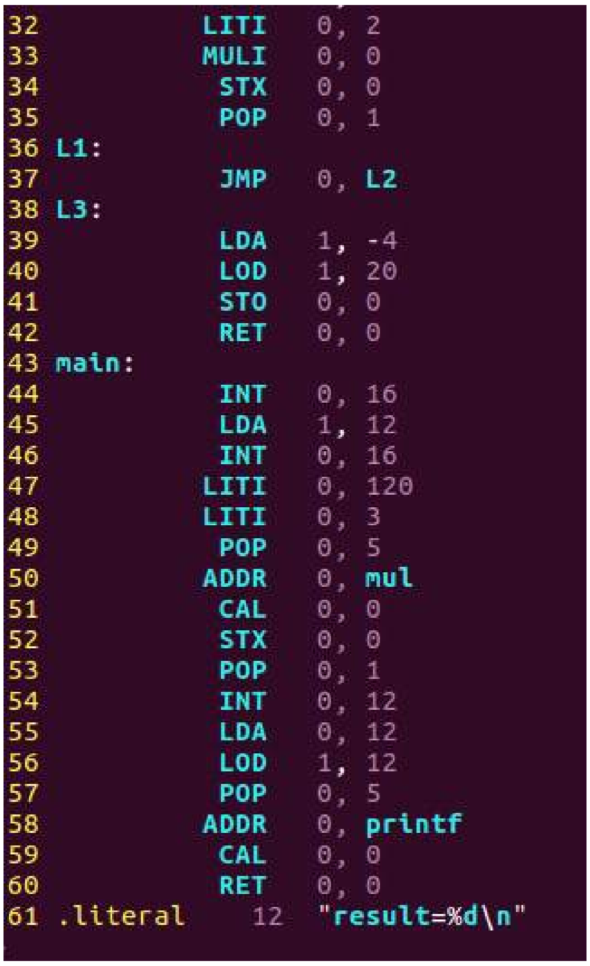
15) test15.c



```c
int mul(int a, int b){
    int result;
    result=0;
    while(a){
        if(a%2)
            result=result+b;
        a=a/2;
        b=b*2;
    }
    return result;
}

void main(){
    int i;
    i=mul(120,3);
    printf("result=%d\n",i);
}
```

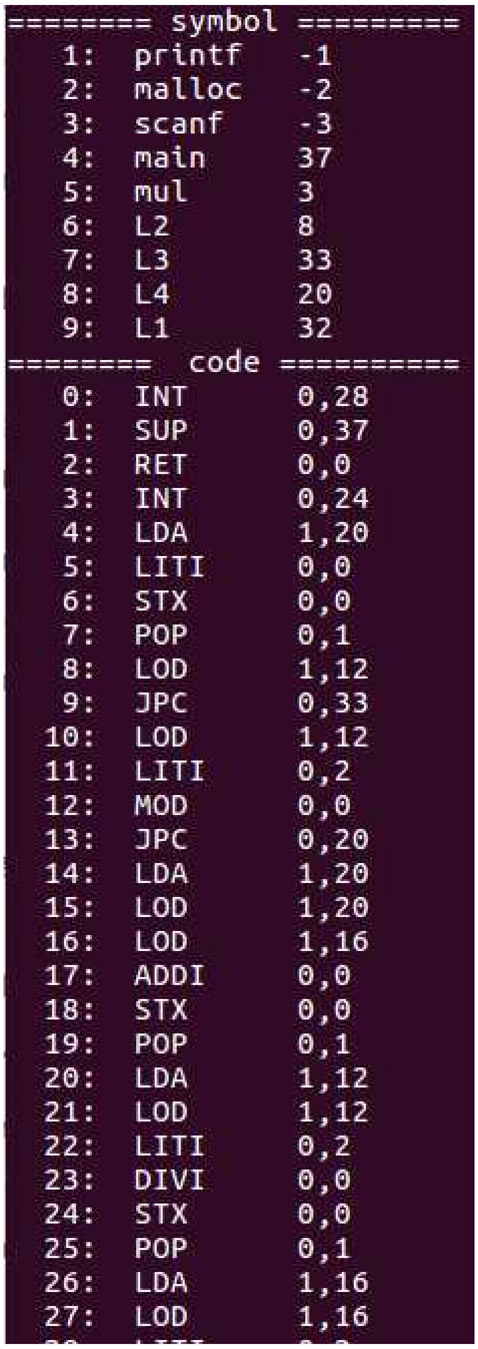
● 컴파일러를 통하여 만들어진 어셈블러 프로그램은 다음과 같다.



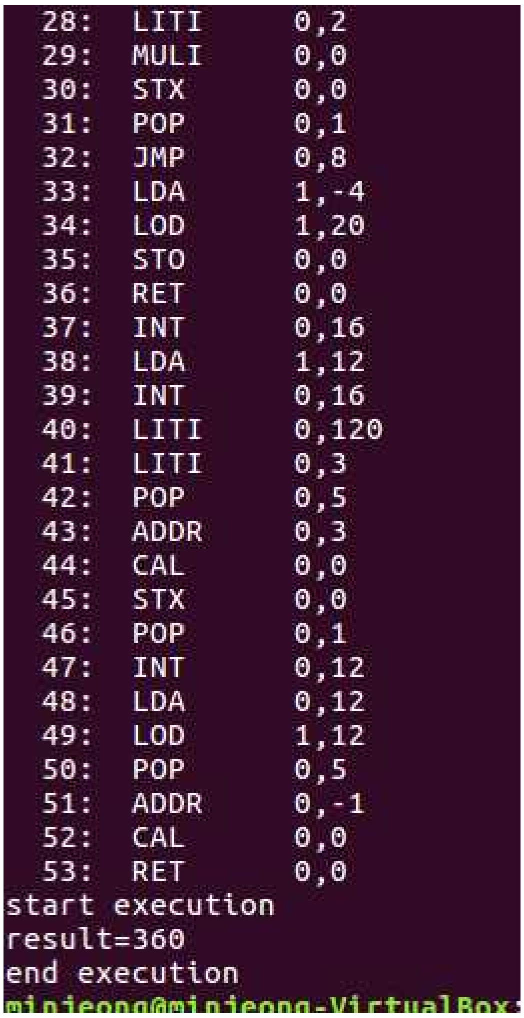


```
        INT     0, 28
        SUP     0, main
        RET     0, 0
mul:
        INT     0, 24
        LDA     1, 20
        LITI    0, 0
        STX     0, 0
        POP     0, 1
L2:
        LOD     1, 12
        JPC     0, L3
        LOD     1, 12
        LITI    0, 2
        MOD     0, 0
        JPC     0, L4
        LDA     1, 20
        LOD     1, 20
        LOD     1, 16
        ADDI    0, 0
        STX     0, 0
        POP     0, 1
L4:
        LDA     1, 12
        LOD     1, 12
        LITI    0, 2
        DIVI    0, 0
        STX     0, 0
        POP     0, 1
        LDA     1, 16
        LOD     1, 16
        LITI    0, 2
        MULI    0, 0
        STX     0, 0
        POP     0, 1
L1:
        JMP     0, L2
L3:
        LDA     1, -4
        LOD     1, 20
        STO     0, 0
        RET     0, 0
main:
        INT     0, 16
        LDA     1, 12
        INT     0, 16
        LITI    0, 120
        LITI    0, 3
        POP     0, 5
        ADDR    0, mul
        CAL     0, 0
        STX     0, 0
        POP     0, 1
        INT     0, 12
        LDA     0, 12
        LOD     1, 12
        POP     0, 5
        ADDR    0, printf
        CAL     0, 0
        RET     0, 0
.literal     12   "result=%d\n"
```

● 어셈블러 프로그램을 시뮬레이터를 이용하여 실행하면 다음과 같은 결과가 나온다.

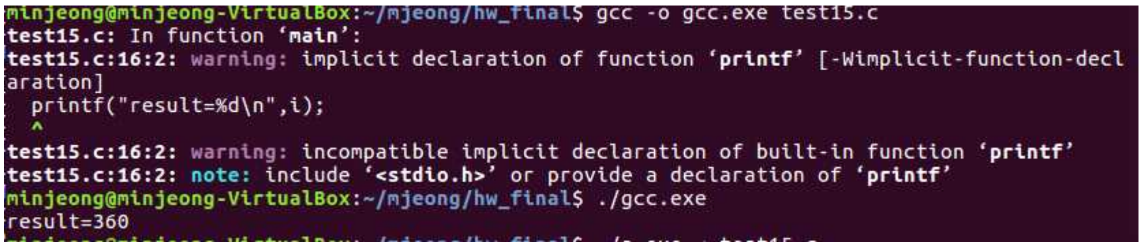


```
======== symbol =========
   1:  printf    -1
   2:  malloc    -2
   3:  scanf     -3
   4:  main      37
   5:  mul       3
   6:  L2        8
   7:  L3        33
   8:  L4        20
   9:  L1        32
========  code ==========
   0:  INT       0,28
   1:  SUP       0,37
   2:  RET       0,0
   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
```



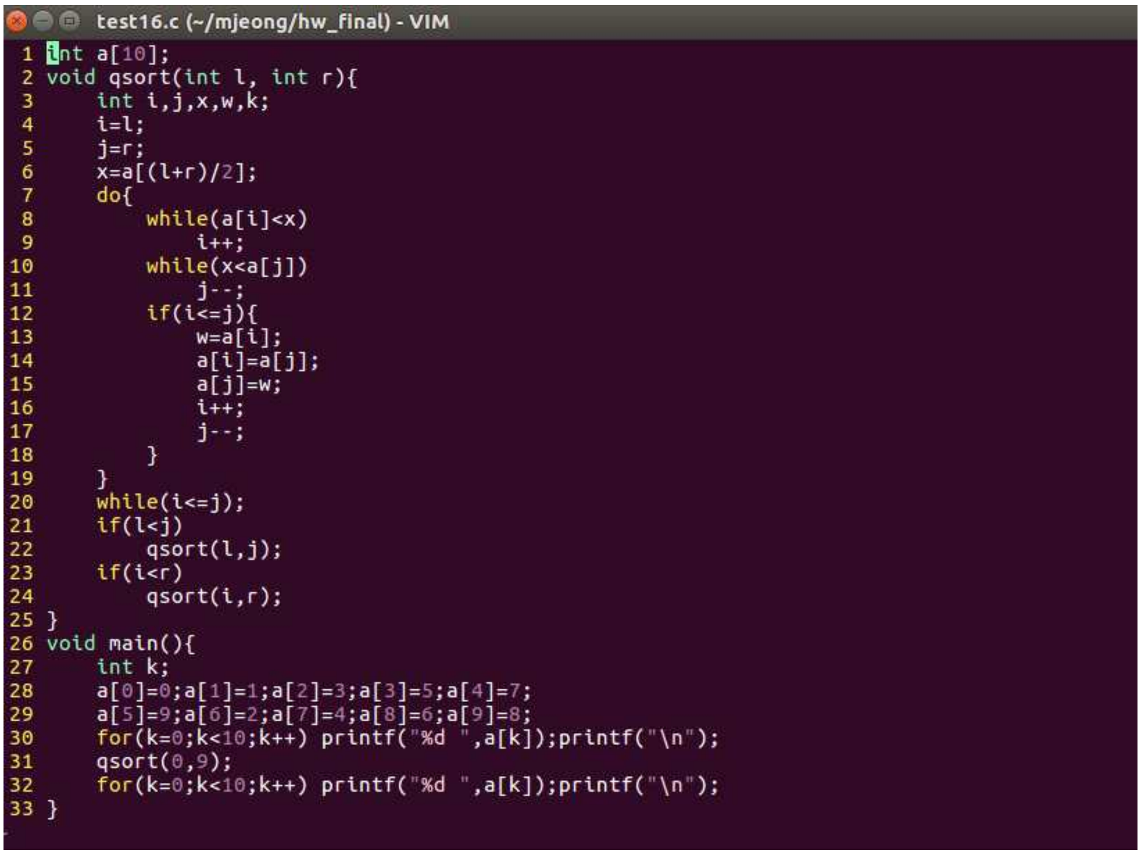
```
  28:  LITI      0,2
  29:  MULI      0,0
  30:  STX       0,0
  31:  POP       0,1
  32:  JMP       0,8
  33:  LDA       1,-4
  34:  LOD       1,20
  35:  STO       0,0
  36:  RET       0,0
  37:  INT       0,16
  38:  LDA       1,12
  39:  INT       0,16
  40:  LITI      0,120
  41:  LITI      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
```

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



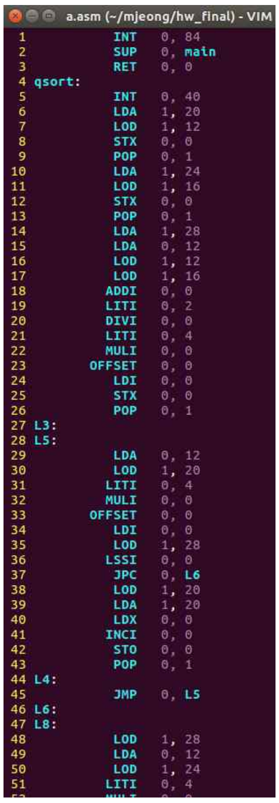
```
minjeong@minjeong-VirtualBox:~/mjeong/hw_final$ gcc -o gcc.exe test15.c
test15.c: In function ‘main’:
test15.c:16:2: warning: implicit declaration of function ‘printf’ [-Wimplicit-function-decl
aration]
  printf("result=%d\n",i);
  ^
test15.c:16:2: warning: incompatible implicit declaration of built-in 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
```

16) test16.c

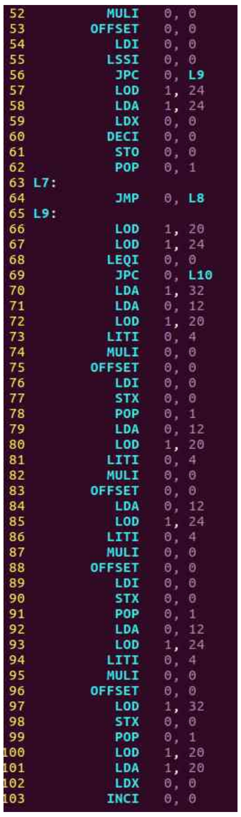


```c
int a[10];
void qsort(int l, int r){
    int i,j,x,w,k;
    i=l;
    j=r;
    x=a[(l+r)/2];
    do{
        while(a[i]<x)
            i++;
        while(x<a[j])
            j--;
        if(i<=j){
            w=a[i];
            a[i]=a[j];
            a[j]=w;
            i++;
            j--;
        }
    }
    while(i<=j);
    if(l<j)
        qsort(l,j);
    if(i<r)
        qsort(i,r);
}
void main(){
    int k;
    a[0]=0;a[1]=1;a[2]=3;a[3]=5;a[4]=7;
    a[5]=9;a[6]=2;a[7]=4;a[8]=6;a[9]=8;
    for(k=0;k<10;k++) printf("%d ",a[k]);printf("\n");
    qsort(0,9);
    for(k=0;k<10;k++) printf("%d ",a[k]);printf("\n");
}
```

● 컴파일러를 통하여 만들어진 어셈블러 프로그램은 다음과 같다.

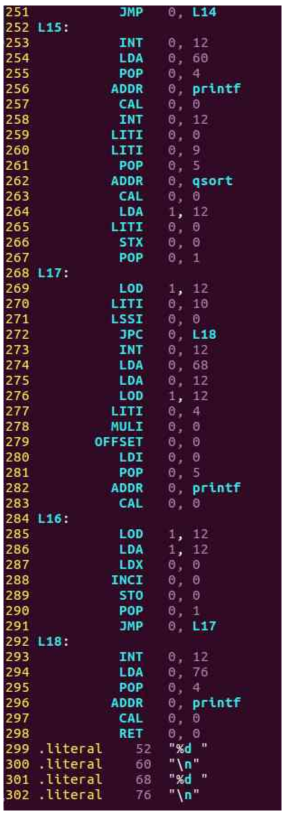


```
          INT     0, 84
          SUP     0, main
          RET     0, 0
qsort:
          INT     0, 40
          LDA     1, 20
          LOD     1, 12
          STX     0, 0
          POP     0, 1
          LDA     1, 24
          LOD     1, 16
          STX     0, 0
          POP     0, 1
          LDA     1, 28
          LDA     0, 12
          LOD     1, 12
          LOD     1, 16
         ADDI     0, 0
         LITI     0, 2
         DIVI     0, 0
         LITI     0, 4
         MULI     0, 0
       OFFSET     0, 0
          LDI     0, 0
          STX     0, 0
          POP     0, 1
L3:
L5:
          LDA     0, 12
          LOD     1, 20
         LITI     0, 4
         MULI     0, 0
       OFFSET     0, 0
          LDI     0, 0
          LOD     1, 28
         LSSI     0, 0
          JPC     0, L6
          LOD     1, 20
          LDA     1, 20
          LDX     0, 0
         INCI     0, 0
          STO     0, 0
          POP     0, 1
L4:
          JMP     0, L5
L6:
L8:
          LOD     1, 28
          LDA     0, 12
          LOD     1, 24
         LITI     0, 4
```



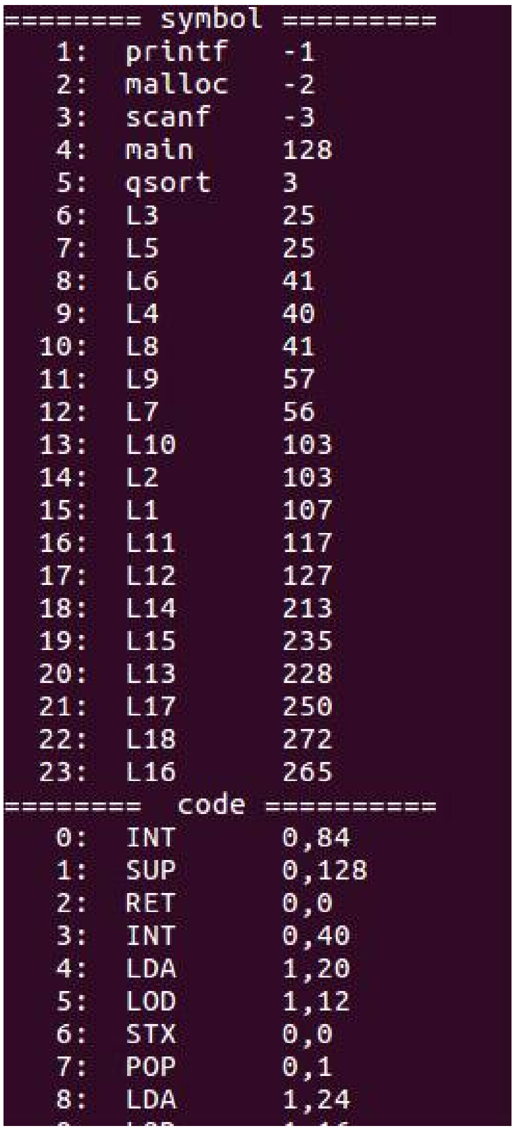
```
         MULI     0, 0
       OFFSET     0, 0
          LDI     0, 0
         LSSI     0, 0
          JPC     0, L9
          LOD     1, 24
          LDA     1, 24
          LDX     0, 0
         DECI     0, 0
          STO     0, 0
          POP     0, 1
L7:
          JMP     0, L8
L9:
          LOD     1, 20
          LOD     1, 24
         LEQI     0, 0
          JPC     0, L10
          LDA     1, 32
          LDA     0, 12
          LOD     1, 20
         LITI     0, 4
         MULI     0, 0
       OFFSET     0, 0
          LDI     0, 0
          STX     0, 0
          POP     0, 1
          LDA     0, 12
          LOD     1, 20
         LITI     0, 4
         MULI     0, 0
       OFFSET     0, 0
          LDA     0, 12
          LOD     1, 24
         LITI     0, 4
         MULI     0, 0
       OFFSET     0, 0
          LDI     0, 0
          STX     0, 0
          POP     0, 1
          LDA     0, 12
          LOD     1, 24
         LITI     0, 4
         MULI     0, 0
       OFFSET     0, 0
          LOD     1, 32
          STX     0, 0
          POP     0, 1
          LOD     1, 20
          LDA     1, 20
          LDX     0, 0
         INCI     0, 0
```

(생략)



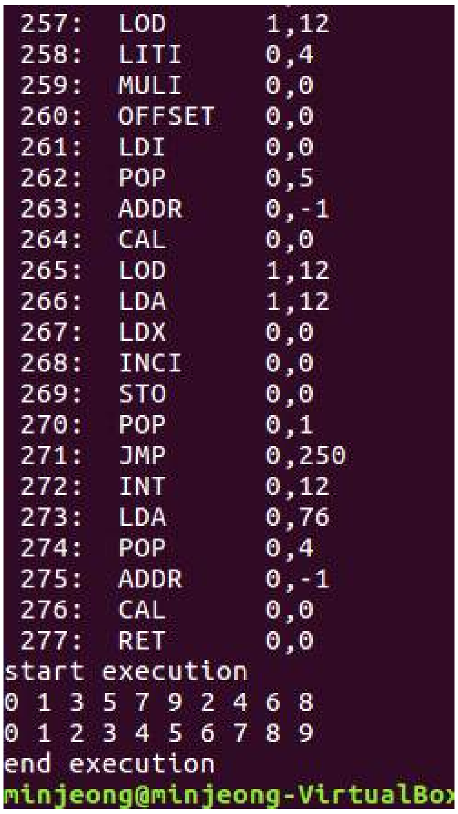
```
          JMP     0, L14
L15:
          INT     0, 12
          LDA     0, 60
          POP     0, 4
         ADDR     0, printf
          CAL     0, 0
          INT     0, 12
         LITI     0, 0
         LITI     0, 9
          POP     0, 5
         ADDR     0, qsort
          CAL     0, 0
          LDA     1, 12
         LITI     0, 0
          STX     0, 0
          POP     0, 1
L17:
          LOD     1, 12
         LITI     0, 10
         LSSI     0, 0
          JPC     0, L18
          INT     0, 12
          LDA     0, 68
          LDA     0, 12
          LOD     1, 12
         LITI     0, 4
         MULI     0, 0
       OFFSET     0, 0
          LDI     0, 0
          POP     0, 5
         ADDR     0, printf
          CAL     0, 0
L16:
          LOD     1, 12
          LDA     1, 12
          LDX     0, 0
         INCI     0, 0
          STO     0, 0
          POP     0, 1
          JMP     0, L17
L18:
          INT     0, 12
          LDA     0, 76
          POP     0, 4
         ADDR     0, printf
          CAL     0, 0
          RET     0, 0
.literal     52   "%d "
.literal     60   "\n"
.literal     68   "%d "
.literal     76   "\n"
```

● 어셈블러 프로그램을 시뮬레이터를 이용하여 실행하면 다음과 같은 결과가 나온다.



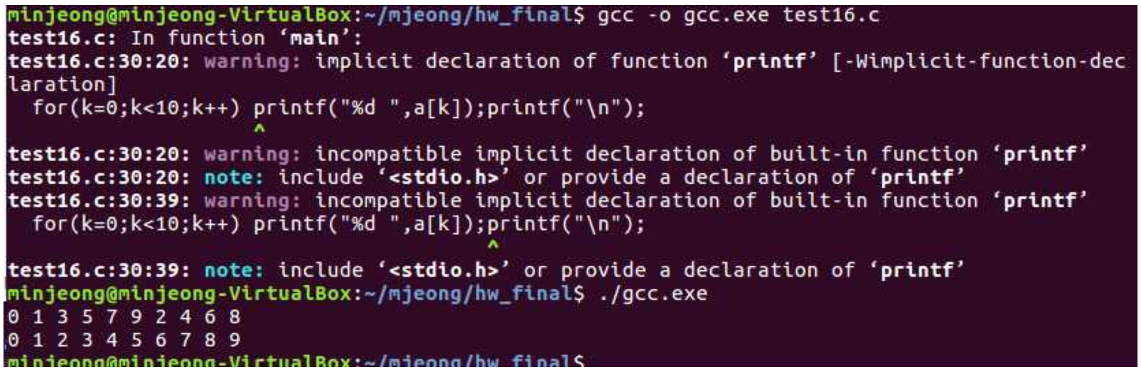
```
======== symbol =========
   1:  printf    -1
   2:  malloc    -2
   3:  scanf     -3
   4:  main      128
   5:  qsort     3
   6:  L3        25
   7:  L5        25
   8:  L6        41
   9:  L4        40
  10:  L8        41
  11:  L9        57
  12:  L7        56
  13:  L10       103
  14:  L2        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-dec
laration]
  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, l1);
        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, l1);
        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);
}
```