Xilinx Zynq FPGA, TI DSP, MCU 기반의 프로그래밍 및 회로 설계 전문가 과정

강사 – Innova Lee (이상훈)
gcccompil3r@gmail.com
학생-김민주
alswnqodrl@naver.com

01

1. 이것이 없으면 사실상 C 언어를 사용할 수 없다. C 언어에서 함수를 호출하기 위해서도 이것은 반드시 필요하다. 이와 같은 이유로 운영체제의 부팅 코드에서도 이것을 설정하는 작업이 진행되는데 이것은 무엇일까 ?

스택이다.
register의 장점은 속도가 빠르나 용량이 작다는 점인데, 이런 점 때문에 임시로 메모리에 스택을 쌓아 저장했다가(입력하면 값이 작아지니까 꺼내쓰기 좋음) 구동한 후 해제해해주는 과정을 취하기 때문이다.
때문에 스택이 없으면 주소가 있어도 아파트가 없는 것과 같은 상황이 펼쳐지기 때문에 c언어 구동이 불가능하다.

02

2. 배열에 아래와 같은 정보들이 들어있다. 여기서 가장 빈도수가 높은 3 개의 숫자를 찾아 출력하시오!
함수에서 이 작업을 한 번에 찾을 수 있도록 하시오.
(찾는 작업을 여러번 분할하지 말란 뜻임)

```
alswnqodrl@alswnqodrl-900X3K:~$ gcc test2.c
alswnqodrl@alswnqodrl-900X3K:~$ ./a.out
1 4 2 7 4 4 7 8 10 4 2 6 1 3 1 9 4 7 4 2 6 4 7 7 8 8 7 3 8 7 8 8
9 10 5 8 8 2
Segmentation fault (core dumped)
alswnqodrl@alswnqodrl-900X3K:~$ vi test2.c
alswnqodrl@alswnqodrl-900X3K:~$ gcc test2.c
alswnqodrl@alswnqodrl-900X3K:~$ ./a.out
10 7 10 6 2 4 3 9 7 8 5 2 4 2 6 4 4 10 7 3 9 6 8 6 3 7 2 9 2 1 1
 3 4 8 9 5 10 10
난수-빈도수
 1 - 7
 2 - 11
 3 - 13
 4 - 11
 5 - 13
 6 - 7
 7 - 11
 8 - 8
 9 - 7
10 - 12
가장 많은 빈도수의 난수 : 3

alswnqodrl@alswnqodrl-900X3K:~$ cat test2.c


    #include <stdio.h>

    #include <stdlib.h>

    #include <time.h>

    void getdata(int rands)

    {//난수를 발생시킨다.

       srand(time(NULL));

    }

    void makeFreq( int rands, int freq[] )

    {// 빈도수를 계산한다.

       int i,n;

       //랜덤한 만큼 반복한다.

       for(i =0; i<rands; i++)

       {//랜덤 숫자가 출력된다.

          n = rand()%10+1;

          printf("%d ",n);

          freq[n-1]++;
```

```
          printf("%d ",n);

          freq[n-1]++;

       }

    }

    void MAX( int freq[] )

    {

       int i, imax, max_num = 0;

       // 빈도수가 가장 많은 것을 찾는다.

       for( i=0; i<10; i++ )

       {

          if( freq[i] > max_num )

          {

             max_num = freq[i];

             imax = i;

          }

       }

       // 빈도수를 출력한다.

       printf("난수-빈도수\n");

       for( i=0; i<10; i++ )

       printf("%2d - %d\n", i+1, freq[i] );

       printf("가장 많은 빈도수의 난수 : %d\n", imax+1 );

    }

    int main(void)

    {//메인 함수 숫자들을 랜덤 시켜서

       // 배열에 저장시킨다.

       int freq[10] = {0};

       // 난수함수를 초기화 한다.

       getdata(100);
```

```
       // 난수함수를 초기화 한다.

       getdata(100);

       // 난수를 100번 발생시킨다.

       makeFreq( 100, freq );

       printf("\n");

       // 가장 많은 빈도수가 가장 많은 것을 출력한다.

       MAX(freq);

       printf("\n");

       return 0;

    }


lswnqodrl@alswnqodrl-900X3K:~$
```

03

3. 12 비트 ADC 를 가지고 있는 장치가 있다.
   보드는 12 V 로 동작하고 있고 ADC 는 -5 ~ 5 V 로 동작한다.
   ADC 에서 읽은 값이 2077 일 때
   이 신호를 디지털 관점에서 재해석하도록 프로그래밍 한다.

디지털 회로(Logic)회로는 단순히 0,1밖에 존재하지 않는데, 0,1은 IC마다 다르지만 대체로 5V를 주로 사용한다. 때문에 외부에서 0~수십V의 신호를 인식하고 0~수십V의 전압을 만들수 있도록 해주는 IC인 ADC 또는 DAC가 필요한데, 이 중ADC는 아날로그 디지털 컨버터 이고, DAC는 디지털 아날로그 컨버터이다. 즉 아날로그 신호를 디지털로 바꾸거나 디지털 신호를 아날로그 신호로 바꾸어주는 중간 장치역할이다.
즉 외부에서 온도값을 읽을때는 ADC를 통해서 몇도인지 알아내고 그것을 설정값과 비교하여 그에 따라 DAC을 통해 아날로그 신호를 내보낸 후 필요하다면 증폭하여 수십V의 전압을 만들어낼수 있는데 어떤 것으로 전압을 증폭하느냐.. 그것은 수많은 회로가 있습니다 .전압전류에 따라 OPAMP+TR로 구성할수도 있고,이런 회로처럼 LM317이란 가변정전압레귤레이터를 이용해 필요한 전압을 만들기도 한다.
위의 경우 보드가 12V로 동작, ADC가 -5~5v(즉, 절댓값이 5), ADC전달값이 2077이므로 이는
2^(12-1) + 2*12+5 = 2077 의 신호로 해석할 수 있다.

04

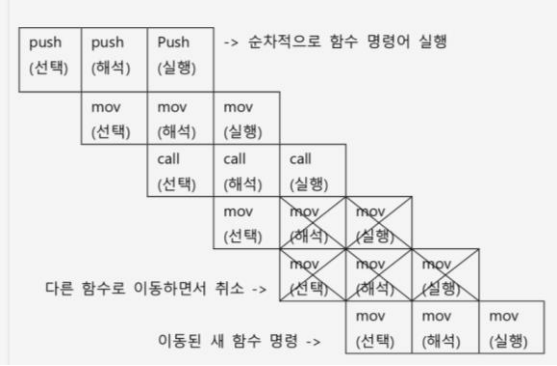
14. 전세계 각지의 천재들이 모여서 개발하는 리눅스 운영체제 코드에는
엄청나게 많은 양의 goto 가 사용되고 있다.
goto 를 도대체 왜 사용해야만 할까 ?

기본작으로 call이나 jmp와 가ㅌ은 명령어를 CPU Instruction 레벨에서 분기 명령어라고 하고 이들은 파이프라인에 매우 치명적인 손실을 준다.

1.fetch-실행할 명력어를 물어옴
2. decode- 어떤 명령어인지 해석
3. excute- 실제 명령어 실행시킴

파이프 라인이 짧은 것부터 긴 것이 5단계에서 수십 단계까지도 구성될 수 있는데 이때 위와 같은 분기 명령어는 3clock을 소요하고 또다시 3단계를 버려야 한다. 여러번의 단계 * 분기횟수만큼 cpu clock을 낭비하게 되는데, 이때 goto의 필요성이 설명된다.

goto를 사용하게 되면여러번의 분기명령어 사용대신 jmp 단 한 번의 사용으로 끝나기 때문에, 분기명령어를 여러번 사용하여 많은 파이프라인 손실을 입지 않을 수 있다.



05

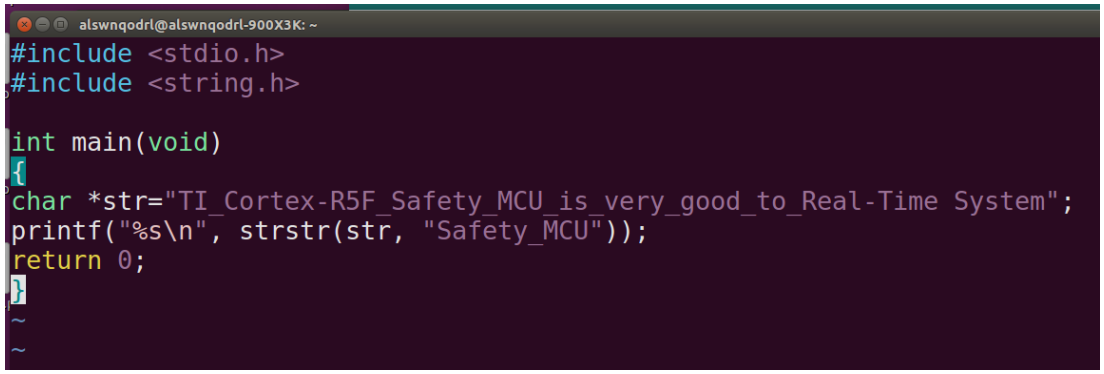
5. 포인터 크기에 대해 알
고 있는대로 기술하시오.

8비트 시스템의 경우 1바이트
16비트의 경우 2바이트
32비트의 경우 4바이트
64비트의 경우 8바이트
=>컴퓨터의 산술 연산이 ALU에 의 존적이기 때문이다.
ALU연산은 범용 레지스터에 종속적이고 컴퓨터가 64비트라는 의미는 이들이 64비트로 구성되었음을 의미한다.
변수의 정의는 메모리에 정보를 저장하는 공간이써고, 포인터의 정의는 메모리에 주소를 저장하는 공간이다. 64비트로 표현할 수 있는 최대값과 최소값을 저장할 대 포인터의 크기가 작으면 표현할 수 없기때문에 최대치인 64비트인 8바이트가 포인터의 크기가 된 것이다.

06

6. TI Cortex-R5F Safety MCU is very good to Real-Time System.

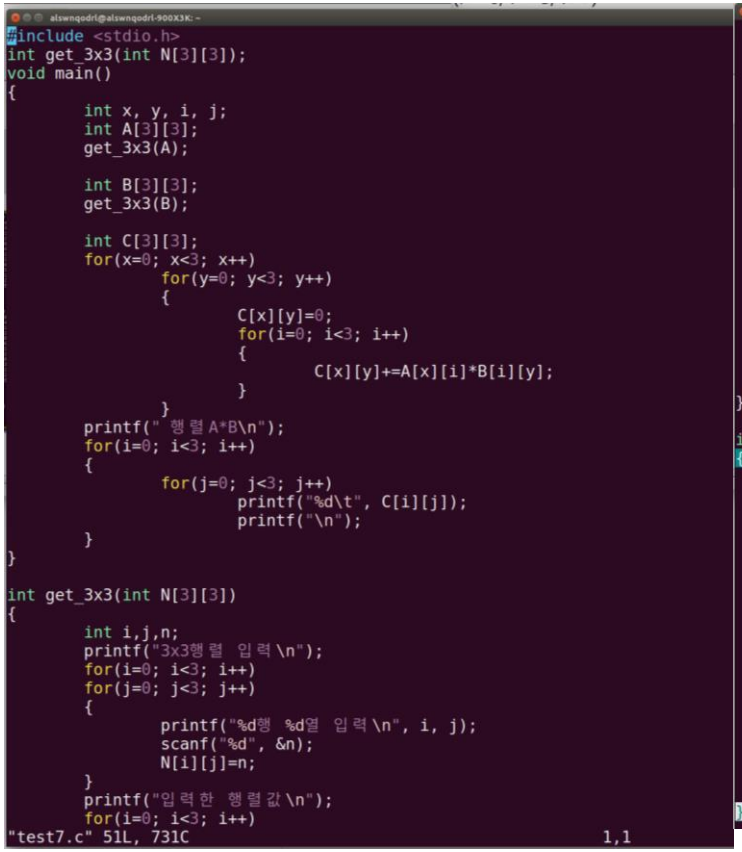
위의 문장에서 Safety MCU 가 몇 번째 부터
시작하는지 찾아내도록 프로그래밍 해보자.
(이정도는 가볍게 해야 파싱 같은것도 쉽게 할 수 있다)




```
alswnqodrl@alswnqodrl-900X3K:~$ vi test6.c
alswnqodrl@alswnqodrl-900X3K:~$ gcc test6.c
alswnqodrl@alswnqodrl-900X3K:~$ ./a.out
Safety_MCU_is_very_good_to_Real-Time System
```

07

7. 이중 배열을 함수의 인자로 입력 받아 3 by 3 행렬의 곱셈을 구현하시오.



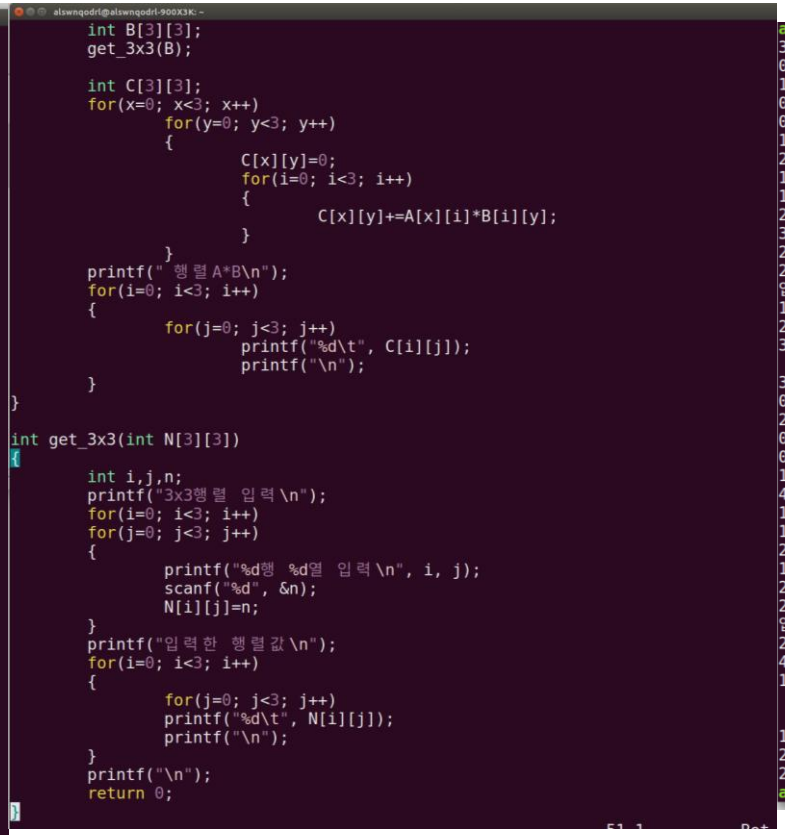
```c
#include <stdio.h>
int get_3x3(int N[3][3]);
void main()
{
        int x, y, i, j;
        int A[3][3];
        get_3x3(A);

        int B[3][3];
        get_3x3(B);

        int C[3][3];
        for(x=0; x<3; x++)
                for(y=0; y<3; y++)
                {
                        C[x][y]=0;
                        for(i=0; i<3; i++)
                        {
                                C[x][y]+=A[x][i]*B[i][y];
                        }
                }
        printf(" 행렬A*B\n");
        for(i=0; i<3; i++)
        {
                for(j=0; j<3; j++)
                        printf("%d\t", C[i][j]);
                        printf("\n");
        }
}

int get_3x3(int N[3][3])
{
        int i,j,n;
        printf("3x3행렬 입력\n");
        for(i=0; i<3; i++)
        for(j=0; j<3; j++)
        {
                printf("%d행 %d열 입력\n", i, j);
                scanf("%d", &n);
                N[i][j]=n;
        }
        printf("입력한 행렬값\n");
        for(i=0; i<3; i++)
```

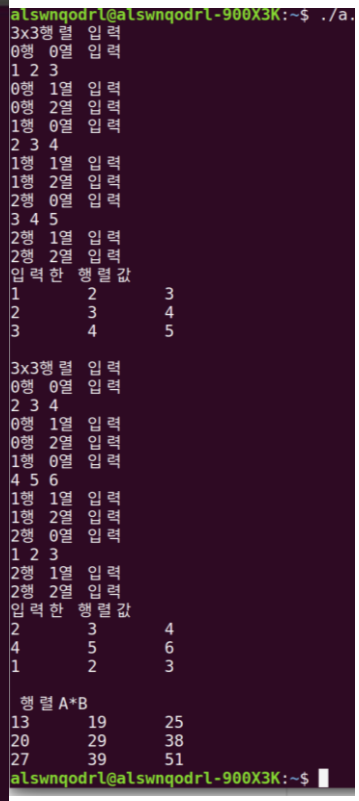
"test7.c" 51L, 731C 1,1



```c
        int B[3][3];
        get_3x3(B);

        int C[3][3];
        for(x=0; x<3; x++)
                for(y=0; y<3; y++)
                {
                        C[x][y]=0;
                        for(i=0; i<3; i++)
                        {
                                C[x][y]+=A[x][i]*B[i][y];
                        }
                }
        printf(" 행렬A*B\n");
        for(i=0; i<3; i++)
        {
                for(j=0; j<3; j++)
                        printf("%d\t", C[i][j]);
                        printf("\n");
        }
}

int get_3x3(int N[3][3])
{
        int i,j,n;
        printf("3x3행렬 입력\n");
        for(i=0; i<3; i++)
        for(j=0; j<3; j++)
        {
                printf("%d행 %d열 입력\n", i, j);
                scanf("%d", &n);
                N[i][j]=n;
        }
        printf("입력한 행렬값\n");
        for(i=0; i<3; i++)
        {
                for(j=0; j<3; j++)
                printf("%d\t", N[i][j]);
                printf("\n");
        }
        printf("\n");
        return 0;
}
```



```
alswnqodrl@alswnqodrl-900X3K:~$ ./a.
3x3행렬 입력
0행 0열 입력
1 2 3
0행 1열 입력
0행 2열 입력
1행 0열 입력
2 3 4
1행 1열 입력
1행 2열 입력
2행 0열 입력
3 4 5
2행 1열 입력
2행 2열 입력
입력한 행렬값
1       2       3
2       3       4
3       4       5

3x3행렬 입력
0행 0열 입력
2 3 4
0행 1열 입력
0행 2열 입력
1행 0열 입력
4 5 6
1행 1열 입력
1행 2열 입력
2행 0열 입력
1 2 3
2행 1열 입력
2행 2열 입력
입력한 행렬값
2       3       4
4       5       6
1       2       3

 행렬A*B
13      19      25
20      29      38
27      39      51
alswnqodrl@alswnqodrl-900X3K:~$
```

08

8. void (* signal(int signum, void (*
handler)(int)))(int) 의 프로토타입을 기술하시오.

void (* signal(int signum, void (* handler)(int)))(int)

1. 리턴

void(*)(int)

2. 이름

signal

3. 인자

(1) Int signum
(2) void(*)(int)

09

9. 함수 포인터를 반환하고 함수 포인터를 인자로 취하는 함수의 주소를 반환하고
   인자로 int 2 개를 취하는 함수를 작성하도록 한다.
   (프로토타입이 각개 다를 수 있으므로 프로토타입을 주석으로 기술하도록 한다)

```c
#include <stdio.h>

int (*p_test_main(float(*p)(int, int)))(float, double, int);
float p_test1(int n1, int n2);
int p_test2(float n1, double n2, int n3);


int(*(*p_test_main2(float(*p)(int, double)))(int))(int, int);
int(*subp1(int n))(int, int);
float p2(int n1, double n2);
int p1(int n1, int n2);

int(*p_test_main(float(*p)(int, int)))(float, double, int)
{
        float res;
        res=p(4, 3);
        printf("p1 res=%f\n", res);
        return p_test2;
}

float p_test1(int n1, int n2)
{
        return(n1+n2)*0.2345;
}

int p_test2(float n1, double n2, int n3)
{
        return(n1+n2+n3)/0.3;
}
```

```c
int(*(*p_test_main2(float(*p)(int, double)))(int))(int, int)
{
        float res;
        res=p(3, 7.7);
        printf("res=%f\n", res);
        return subp1;
}

int p1(int n1, int n2)
{
        return n1+n2;
}

int(*subp1(int n))(int, int)
{
        printf("n=%d\n", n);
        return p1;
}

float p2(int n1, double n2)
{
        return n1*n2;
}

int main()
{
        int res=p_test_main(p_test1)(3.7, 2.4, 7);
        printf("p_test res=%d\n", res);
        res=p_test_main2(p2)(3)(7, 3);
        printf("res2=%d\n", res);
        return 0;
}
```

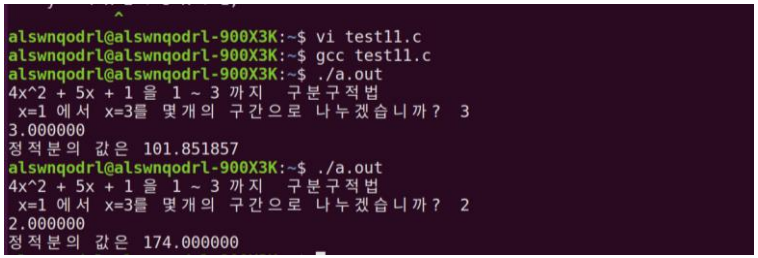
```
alswnqodrl@alswnqodrl-900X3K:~$ vi test9.c
alswnqodrl@alswnqodrl-900X3K:~$ gcc test9.c
alswnqodrl@alswnqodrl-900X3K:~$ ./a.out
p1 res=1.641500
p_test res=43
res=23.100000
n=3
res2=10
alswnqodrl@alswnqodrl-900X3K:~$
```

10

10. 파이프라인이 깨지는 경우 어떠한 이점이 있는지 기술하시오.

11

11. 4x^2 + 5x + 1 을 1 ~ 3 까지 정적분하는 프로그램을 구현하도록 한다.




```c
#include <stdio.h>

int main()
{
 double x=0;
 double y=0;
 double sum=0;
 float n;

   printf("4x^2 + 5x + 1 을 1 ~ 3 까지  구분구적법\n");
   printf(" x=1 에서 x=3를 몇개의 구간으로 나누겠습니까?  ");
   scanf("%f",&n);

       printf("%f\n",n);

        while(x<=3)
       {
        x = x+(2/n);
        y = 4*x*x + 5*x + 1;
        sum = sum + (y*2/n);
 }

 printf("정적분의 값은 %f\n",sum);

 return 0;

}
```

12

함수 포인터를 활용하여 float 형 3 by 3 행렬의 덧셈과 int 형 3 by 3 행렬의 덧셈을 구현하도록 하시오.

```c
#include <stdio.h>
int get_3x3(int N[3][3]);
void main()
{
	int x, y, i, j;
	int A[3][3];
	get_3x3(A);

	int B[3][3];
	get_3x3(B);

	int C[3][3];
	for(x=0; x<3; x++)
		for(y=0; y<3; y++)
		{
			C[x][y]=0;
			for(i=0; i<3; i++)
			{
				C[x][y]+=A[x][i]*B[i][y];
			}
		}
	printf(" 행렬A*B\n");
	for(i=0; i<3; i++)
	{
		for(j=0; j<3; j++)
			printf("%d\t", C[i][j]);
			printf("\n");
	}
}

int get_3x3(int N[3][3])
{
	int i,j,n;
	printf("3x3행렬 입력\n");
	for(i=0; i<3; i++)
	for(j=0; j<3; j++)
	{
		printf("%d행 %d열 입력\n", i, j);
		scanf("%d", &n);
		N[i][j]=n;
	}
	printf("입력한 행렬값\n");
	for(i=0; i<3; i++)
```

"test7.c" 51L, 731C 1,1

```c
	int B[3][3];
	get_3x3(B);

	int C[3][3];
	for(x=0; x<3; x++)
		for(y=0; y<3; y++)
		{
			C[x][y]=0;
			for(i=0; i<3; i++)
			{
				C[x][y]+=A[x][i]*B[i][y];
			}
		}
	printf(" 행렬A*B\n");
	for(i=0; i<3; i++)
	{
		for(j=0; j<3; j++)
			printf("%d\t", C[i][j]);
			printf("\n");
	}
}

int get_3x3(int N[3][3])
{
	int i,j,n;
	printf("3x3행렬 입력\n");
	for(i=0; i<3; i++)
	for(j=0; j<3; j++)
	{
		printf("%d행 %d열 입력\n", i, j);
		scanf("%d", &n);
		N[i][j]=n;
	}
	printf("입력한 행렬값\n");
	for(i=0; i<3; i++)
	{
		for(j=0; j<3; j++)
		printf("%d\t", N[i][j]);
		printf("\n");
	}
	printf("\n");
	return 0;
}
```

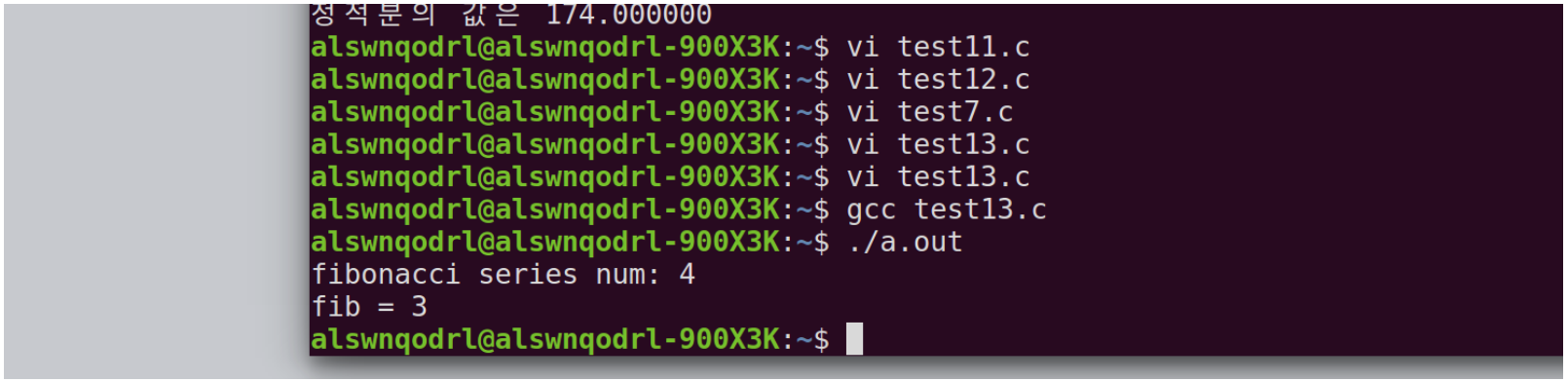
```
alswnqodrl@alswnqodrl-900X3K:~$ ./a.
3x3행렬 입력
0행 0열 입력
1 2 3
0행 1열 입력
0행 2열 입력
1행 0열 입력
2 3 4
1행 1열 입력
1행 2열 입력
2행 0열 입력
3 4 5
2행 1열 입력
2행 2열 입력
입력한 행렬값
1	2	3
2	3	4
3	4	5

3x3행렬 입력
0행 0열 입력
2 3 4
0행 1열 입력
0행 2열 입력
1행 0열 입력
4 5 6
1행 1열 입력
1행 2열 입력
2행 0열 입력
1 2 3
2행 1열 입력
2행 2열 입력
입력한 행렬값
2	3	4
4	5	6
1	2	3

 행렬A*B
13	19	25
20	29	38
27	39	51
alswnqodrl@alswnqodrl-900X3K:~$
```

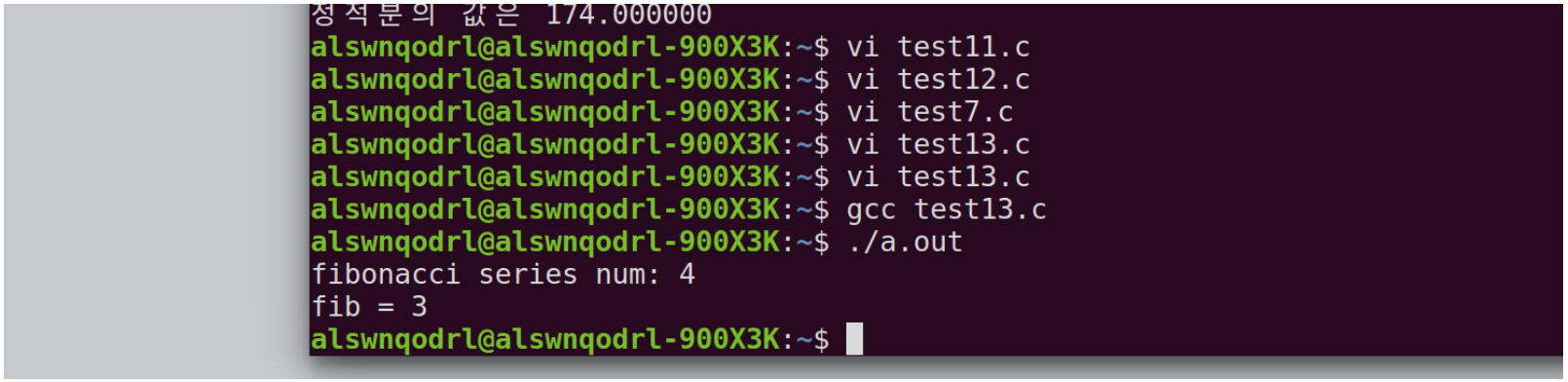
13

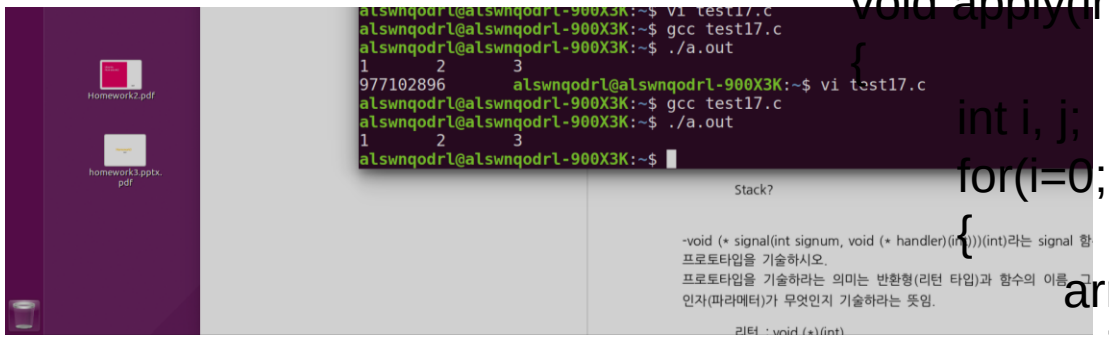
1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ... 형태로 숫자가 진행된다.
1 ~ 27 번째까지의 수들로 홀수들의 합을 하고 짝수들의 합을 구한다.
홀수들의 합 - 짝수들의 합의 결과를 출력하시오.



14

1, 4, 5, 9, 14, 23, 37, 60, 97, ... 형태로 숫자가 진행된다.
23번째 숫자는 무엇일까 ?
(프로그래밍 하시오)






```c
#include <stdio.h>

int func(int n)
{
	return n*n;
}

void apply(int arr[2][3], int (*p)(int n))
{
	int i, j;
	for(i=0; i<3; i++)
	{
		arr[1][i]=func(arr[0][i]);
		printf("%d\t", arr[0][i]);
	}


	printf("\n");
	for(i<0; i<3; i++)
	printf("%d\t", arr[1][i]);
}

int main(void)
{
```

```
#include <stdio.h>
int main(void)
{
        char str1[5]="AAA";
        char str2[5]="BBB";
        char
str3[]={'A','B','C'};
        char
str4[]={'A','B','C','\0'};

        printf("str1 = %s\n",
str1);
        printf("str2 = %s\n",
str2);
        printf("str3 = %s\n",
str3);
        printf("str4 = %s\n",
str4);

        str1[0]='E';
```

17

```
#include
int func(int n){        return n*n;}
void apply(int arr[2][3], int (*p)(int n)){        int i, j;        for(i=0; i<3; i++)        {                arr[1][i]=func(arr[0][i]);                printf("%d\t", arr[0][i]);        }
        printf("\n");        for(i<0; i<3; i++)        printf("%d\t", arr[1][i]);}
int main(void){        int arr[2][3]={{1,2,3},{1,2,3}};        apply(arr, func);}~
```

18

```
#include int main(void){    char str1[5]="AAA";    char str2[5]="BBB";    char str3[]={'A','B','C'};    char str4[]={'A','B','C','\0'};
    printf("str1 = %s\n", str1);    printf("str2 = %s\n", str2);    printf("str3 = %s\n", str3);    printf("str4 = %s\n", str4);
    str1[0]='E';    str1[1]='H';        printf("str1 = %s\n", str1);        return 0;}
```

다른 것이다.

int (*p)[3]은 인트형 데이터 3개를 저장할 수 있는 배열을 가리키는 포인터 이고
int *p[3]은 포인트 인트형 데이터를 3개 저장할 수 있는 포인트형 배열 p이다.

20

```
#include <stdio.h>

int main()
{
        int x;
        scanf("%d", &x);
        x&=~(134217728-1);
        printf("%d\n", x);
        return 0;
}
~
~
```

21

```
alswnqodrl@alswnqodrl-900X3K: ~
#include <stdio.h>

int main(void)
{
        char ch;
        scanf("%c", &ch);
        printf("%c\n", ch^32);
        return 0;
}
~
~
~
~
```

```
alswnqodrl@alswnqodrl-900X3K:~$ vi test21.c
alswnqodrl@alswnqodrl-900X3K:~$ gcc test21.c
alswnqodrl@alswnqodrl-900X3K:~$ ./a.out
a
A
alswnqodrl@alswnqodrl-900X3K:~$ ./a.out
A
a
alswnqodrl@alswnqodrl-900X3K:~$ vi test21.c
alswnqodrl@alswnqodrl-900X3K:~$ vi test21.c
alswnqodrl@alswnqodrl-900X3K:~$
```

변수란 메모리에 존재하는 정보(주소[포인터])를 저장할 수 있는 공간을 의미한다.

포인터란 주소값을 저장할 수 있는 공간을 의미한다.

함수 포인터란 포인터 안에 특정 함수를 저장해 둔 후 필요할 때 마다 해당 포인터에 접근하여 그 함수를 구동시킬 수 있도록 하는 것이다.

파이프라인은 call, jmp와 같은 분기명령어가 구동될 때 단계*분기횟수 만큼 cpu clock이 낭비되면서 깨지게 된다.
goto가 이런 점을 잡아줄 수 있다.

fast
Rgister
chche
memory
disk
slow

속도는 register가 빠르나 용량이 적고, 용량은 디스크가 가장 많으나 속도가 느림
=> 디스크보다는 메모리를 위주로 많이 사용함

◆ 메모리상의 구조

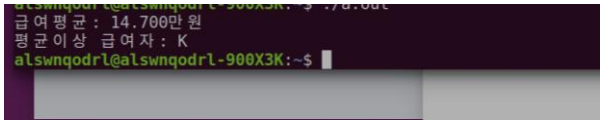
◆ 메모리구조의 정리

	스택 영역 (Stack Area)	데이터 영역 (Data Area)	코드 영역 (Code Area)	힙 영역 (Heap Area)
특징	기본자료형(Int, char, double, void, float, 지역변수) 표준버퍼, 시스템(OS)임시데이터	전역변수, 정적변수 (static)	함수코드(원형부), 함수 구현부	배역, 클래스, 컬렉션, 등의 동적변수 (크기를 정할수 없는 변수)
속도 순위	1	2	3	4

우리가 사용하는 윈도우나 리스 맥 등에서 보는 변수 주소는 진짜가 아니라 임의로 할당된 주소라고 할 수 있다.

29

```
#include <stdio.h>#include <stdlib.h>#include <time.h>
typedef struct Person{
int pay;char name[20];}Person;
int main(void){ srand(time(NULL)); Person P[7]={{0,'K'}}; double ave; int i, sum=0; for(i=0; i<7;
i++) { P[i].pay=rand()%30+1; sum+=P[i].pay; } ave=sum/7.0; printf("급여평균: %.01f00만원\n", ave); printf("평균이상 급
여자: %s\n", P[0].name); return 0;}
```



1) gcc -g 컴파일명.c
2) gcc -g -O0 -o debug 컴파일명.c
3) dgb debug
4) b main
5) r
6) si
7) disas

ctrl+v를 누른 후 위에서 푸터 방향키 아래를 눌러 모두 비쥬얼모드에서 블럭을 씌워준 후 =을 누르면 자동으로 들여쓰기가 되며 보기 좋게 정렬이 된다.

최적화를 방지하기 위해선
gcc -g -O0 -o 컴파일명.c
또는
volatile 키워드를 이용한다.

최적화를 방지하기 위해선
gcc -g -O0 -o 컴파일명.c
또는
volatile 키워드를 이용한다.

34

gdb 를 사용하는 이유를 기술하시오.

코딩했을 때 문법적인 오류는 없으나 논리적인 오류에 빠져서 빠져나오지 못할 때 원인이 되는 것이 무엇인지 기계어를 살피기 위해
시행한다.

36

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>


int main(void){


    int input, a;
    int draw = 0, win = 0;


    //사용자가 질 때 까지 반복
    while (1){
        printf("바위는 1, 가위는 2, 보는 3: ");
        scanf("%d", &input);


//srand()와 time()을 이용해서 1~3까지 난수생성
srand((int)time(NULL));
a = rand() % ((3 - 1) + 1) + 1;


        //비겼을 때
        if (input == a){


            //난수가 바위이면
            if (a == 1){
                printf("당신은 바위, 컴퓨터는 바위 선택, 비겼습니다.");
                draw++;
            }


            //난수가 가위이면
            else if (a == 2){
                printf("당신은 가위, 컴퓨터는 가위 선택, 비겼습니다.");
                draw++;
            }


            //난수가 보자기이면
            else{
                printf("당신은 보자기, 컴퓨터는 보자기 선택, 비겼습니다.");
                draw++;
            }
        }
```

```c
        //사용자가 졌을 때
        else{


            if (input == 1 && a == 3){
                printf("당신은 바위, 컴퓨터는 보자기 선택, 졌습니다.");
                break;
            }


            else if (input == 2 && a == 1){
                printf("당신은 가위, 컴퓨터는 바위 선택, 졌습니다.");
                break;
            }


            else{
                printf("당신은 보자기, 컴퓨터는 가위 선택, 졌습니다.");
                break;
            }
        }


        printf("\n\n");
    }


    printf("\n\n");
    printf("게임의 결과 : %d승, %d무", win, draw);

    return 0;
}
```

```
alswnqodrl@alswnqodrl-900X3K:~$ gcc test36.c
alswnqodrl@alswnqodrl-900X3K:~$ ./a.out
바위는 1, 가위는 2, 보는 3: 1
당신은 보자기, 컴퓨터는 가위 선택, 졌습니다.

게임의 결과 : 0승, 0무alswnqodrl@alswnqodrl-900X3K:~$ ./a.out
바위는 1, 가위는 2, 보는 3: 2
당신은 보자기, 컴퓨터는 가위 선택, 졌습니다.

게임의 결과 : 0승, 0무alswnqodrl@alswnqodrl-900X3K:~$ ./a.out
바위는 1, 가위는 2, 보는 3: 3
당신은 보자기, 컴퓨터는 바위 선택, 이겼습니다.

바위는 1, 가위는 2, 보는 3: 2
당신은 보자기, 컴퓨터는 바위 선택, 이겼습니다.

바위는 1, 가위는 2, 보는 3: 1
당신은 바위, 컴퓨터는 보자기 선택, 졌습니다.

게임의 결과 : 2승, 0무alswnqodrl@alswnqodrl-900X3K:~$
```

40

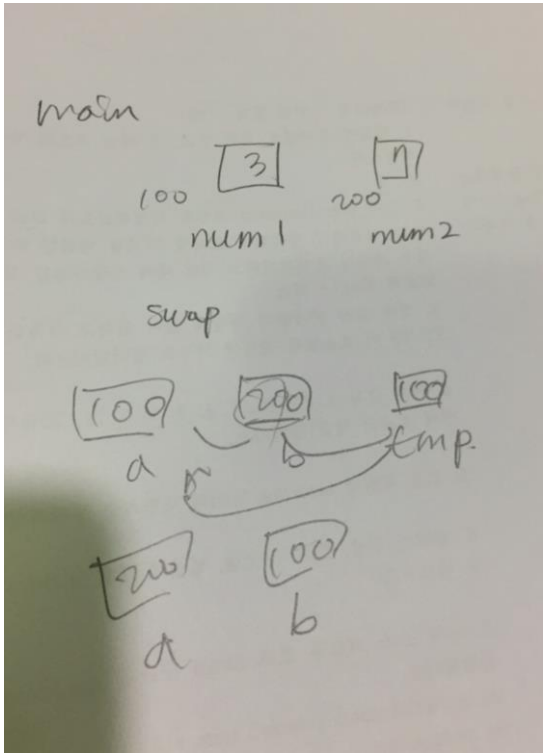
한 달간 수업을 진행하면서 본인이 느낀점을 20 줄 이상 최대한 상세하게 기술하시오.
또한 앞으로에 대한 포부 또한 기술하길 바란다.
그리고 앞으로 어떤 일을 하고 싶은지 상세히 기술하도록 한다.

새로 접하는 내용들이 많아 벅차기도 했지만 많은 일들을 챙겨야하는 상황 속에서도 나름 처음 배우는 과목에 대해 즐겁게 배울 수 있었던 것 같다.
앞으로 조금 더 성실한 자세로 교육에 임하여 교육이 끝날 때 쯤 원하는 회사에 취직하여 이 분야에 대해 자신감 있어지고 싶다.

41

```c
#include <stdio.h>

void swap(int *a, int *b){
    int tmp;
    tmp = *a;
    *a = *b;
    *b = tmp;
}
int main(void){
    int num1 = 3, num2 = 7;
    swap(&num1, &num2);
    return 0;
}
```



a와 b에 저장되는 주소값을 서로 바꿔주는 함수이다.