Xilinx Zynq FPGA,TI DSP, MCU 기반의 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com

학생 – 정한별
hanbulkr@gmail.com

<5일차 내용 정리>

배열이 필요한 이유는?

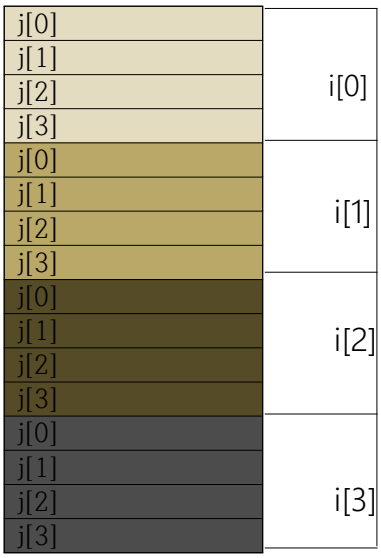
-배열이 필요한 이유는? 100개의 변수 저장시 for을 이용하여 할당한다.
-int sensor[1000]={0}; 이런 식으로 초기화 한다.
-배열은 순차적으로 배열되어 있다.
-배열은 2차원이 아닌 2중 배열, 3중 배열이라 하는게 맞다.
-c 언어 에서는 배열에 차원 개념이 없다.

NULL Character 는 무엇인가?

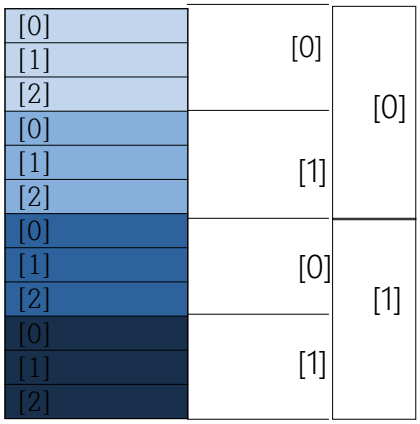
-배열의 크기를 지정해 줄 때, +1, 혹은 +2를 해서 크기를 잡는게 좋다.
(값이 넘어가는 경우가 가끔 생길 때가 있기 때문에)

*단위행렬이 메모리에 어떻게 저장 될까?

(4*4짜리 일 때)



(arr[2][2][3] 배열 일 때)



*위와 같이 배치가 되어 저장 된다. 된다.

배열=(주소)

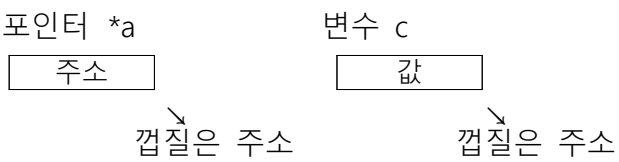
우리가 봤던 주소는 가짜, 그럼 진짜 주소는?

- 1. 가상메모리
- 2.페이징

포인터=(주소)

*포인터를 쓰는 이유는 함수 어디에서든 다른함수의 지역 변수를 사용할 수 있다.
*주소를 저장할 수 있는 변수
*포인터의 크기는 H/w가 몇bit를 가지냐에 따라 다른다.

선언방법
*가리키고 싶은 녀석의 자료형을 적는다.
*그냥 *p라고 하면 포인터의 값을 의미, 해당변수 p가 가리키는 값을 의미.



*포인터변수 a라는 주소는 주소를 저장하고 있다.
*변수 c라는 주소에는 데이터, 값이 들어있다.

* Segmentation Fault 가 나는 이유 ?

우리가 기계어를 보면서 살펴봤던 주소값들이
사실은 전부 가짜 주소라고 말했었다.
이 주소값은 엄밀하게 가상 메모리 주소에 해당하고
운영체제의 Paging 메커니즘을 통해서
실제 물리 메모리의 주소로 변환된다.
(윈도우도 가상 메모리 개념을 베껴서 사용한다)
그렇다면 당연히 맥(유닉스)도 사용함을 알 수 있다.

가상 메모리는 리눅스의 경우
32 비트 버전과 64 비트 버전이 나뉜다.

32 비트 시스템은 2^32 = 4GB 의 가상 메모리 공간을 가짐
여기서 1:3 으로 1 을 커널이 3 을 유저가 가져간다.
1 은 시스템(HW, CPU, SW 각종 주요 자원들)에
관련된 중요한 함수 루틴과 정보들을 관리하게 된다.
3 은 사용자들이 사용하는 정보들로
문제가 생겨도 그다지 치명적이지 않은 정보들로 구성됨

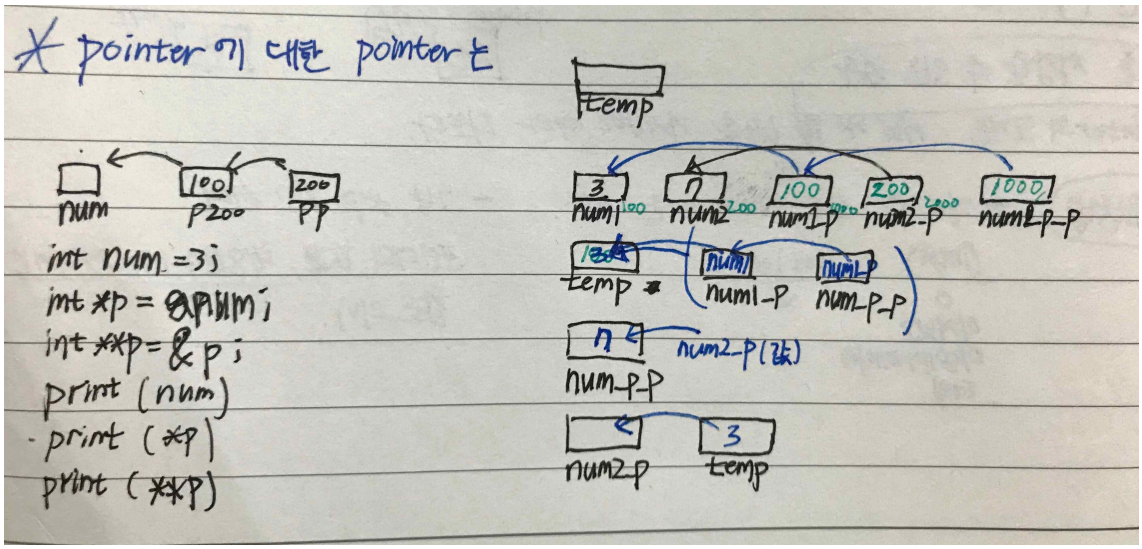
64 비트 시스템은 1:1 로
2^63 승에 해당하는 가상메모리를 각각 가진다.
문제는 변수를 초기화하지 않았을 경우
가지게 되는 쓰레기값이 0xCCCCC...CCC 로 구성됨이다.

32 비트의 경우에도 1:3 경계인 0xC0000000 을 넘어가게됨
64 비트의 경우엔 시작이 C 이므로
이미 1:1 경계를 한참 넘어감

그러므로 접근하면 안되는 메모리 영역에 접근하였기에
엄밀하게는 Page Fault(물리 메모리 할당되지 않음) 가
발생하게 되고 원래는 Interrupt 가 발생해서
Kernel 이 Page Handler(페이지 제어기)가 동작해서
가상 메모리에 대한 Paging 처리를 해주고
실제 물리 메모리를 할당해주는데
문제는 이것이 User 쪽에서 들어온 요청이므로
Kernel 쪽에서 강제로 기각해버리면서
Segmentation Fault 가 발생하는 것이다.

실제 Kernel 쪽에서 들어온 요청일 경우에는
위의 메커니즘에 따라서 물리 메모리를 할당해주게 된다.

- *(&num)에서 '*', '&'는 상충되어 사라진다. 결국 그냥 num 이 된다.
- '*' 는 주소값을 주겠다.
- '&' 는 주소값을 받겠다.



<포인터 배열, 배열 포인터>

int *arr_ptr[3] =(&n1, &n2, &n3); --> (int*)이라는 데이터 타입
의미: int형 타입의 포인터가 3개 있다.

int (*p)[2]=a; --> int *[2]p 형태로 보면 된다.
의미: int형 2개"짜리에 대한 포인터_

ex)

```
int a[2][2] ={{10,20},{30,40}};
int (*p)[2]=a;

for(i= 0; i< 2; i++)
        printf("p[%d] = %d₩n", i, *p[i]);
```

위와 같을 때, 이걸 순서대로 프린트를 하면 10,30 이 나온다.
(그 뒤는 똥값이랑 합쳐져 이상한 크기에 수가 나옴.)

왜 그럴까? 그건 바로 int형의 크기가 2개씩 있기 때문에 8byte씩 움직인다.
그래서 *p[0]은 첫 항인 a[0][0]을 가리키고 *p[1]는 8byte후인 a[1][0]를 나타낸다.

<문제 은행>

http://cafe.naver.com/hestit/79

q1

```
#include <stdio.h>

void arr_print(int num)
{

		if(!(num%2))
			printf("%d ",num);


}


int main(void)
{
		int i=0;
		int t=0;
		int a[t];

		printf("배열의 크기를 입력하시오₩n");
		scanf("%d",&t);

		printf("배열에 들어가 요소를 %d개 입력하시요₩n",t);

		for(i=0;i<t;i++)
		{
				scanf("%d",&a[i]);
		}

		for(i=0;i<t;i++)
				arr_print(a[i]);
		printf("₩n");


		return 0;
}
```

q3

```
#include<stdio.h>

void change(int *arr,int *arr2,int len)
{
        int i=0;
        //int len= sizeof(arr)/4;

        printf("배열의 길이는 몇인가? %d₩n",len);
        for(i=0;i<len;i++){
                        arr2[len-i-1]=arr[i];
        }
}
void print_arr(int *arr2,int len)
{
                int i;
                //int len= sizeof(arr2)/4;

                printf("배열의 길이는 몇인가? %d₩n",len);
                printf("배열의 요소는?₩n");
                for(i =0;i<len;i++)
                                printf("%4d",arr2[i]);
}

int main(void)
{
                int arr[]={3,77,10,7,4,9,1,8,21,33};
                int len= sizeof(arr)/4;
                int arr2[len];
                int i;

                printf("배열의 길이는 몇인가? %d₩n",len);
                change(arr,arr2,len);
                print_arr(arr2,len);
                printf("₩n");

                return 0;
}
```

q4

```
#include<stdio.h>

int change(int *arr,int len)
{
        int i=0;
        int even=0;
        int odd=0;
//      int res=0;

        for(i=0;i<len;i++){
                if(!(arr[i]%2))
                                even+=arr[i];
                else
                                odd+=arr[i];
        }
        return odd*even;
}

int main(void)
{
                int arr[]={3,77,10,7,4,9,1,8,21,33};
                int len= sizeof(arr)/4;
                int i;

                printf(" 결과 = %d₩n",change(arr,len));

                return 0;
}
```

q6.

행렬 덧셈 & 행렬 뺄셈

$$(\mathbf{A} + \mathbf{B})_{ij} = \mathbf{A}_{ij} + \mathbf{B}_{ij}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 3 & 3 \\ 5 & 5 & 7 \\ 7 & 9 & 9 \end{bmatrix}$$

$$(\mathbf{A} - \mathbf{B})_{ij} = \mathbf{A}_{ij} - \mathbf{B}_{ij}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} - \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 3 \\ 3 & 5 & 5 \\ 7 & 7 & 9 \end{bmatrix}$$

행렬 곱셈

$$A \times B = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \times \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 4 & 2 \\ 5 & 10 & 5 \\ 8 & 16 & 8 \end{bmatrix}$$

행렬 나눗셈

행렬의 나눗셈은 우리가 생각하는 것과 조금 다르다. 역행렬이라는 개념이 들어간다.

$\frac{A}{B} = A \cdot B^{-1}$ 이고 $A \cdot B^{-1} \neq B^{-1} \cdot A$ 이다.

http://cafe.naver.com/hestit/104

q2

```
#include<stdio.h>


int main(void)
{
		unsigned int a = 2004016;
		char *p=&a;

		printf("%p ₩n",&a);
		printf("%p ₩n",&a+0x01);
		printf("%p ₩n",&a+0x02);
		printf("%p ₩n",&a+0x03);
		printf("₩n");
		printf("%c ₩n",&a);
		printf("%c ₩n",&a+0x01);
		printf("%c ₩n",&a+0x02);
		printf("%c ₩n",&a+0x03);

		return 0;
}
```

q4

```
#include <stdio.h>

int main(void)
{
		int num1 = 3, num2 =7, num3=5;
		int *temp = NULL;
		int *num1_p = &num1;
		int *num2_p = &num2;
		int *num3_p = &num3;
		int **num_p_p= &num1_p;

		printf("num1_p= %d ₩n", *num1_p);
		printf("num2_p= %d ₩n", *num2_p);
		printf("num3_p= %d ₩n", *num3_p);

		printf("₩n");

		for(;;){
				temp = num1_p;
				num1_p =num2_p;
				num2_p = num3_p;
				num3_p = temp;

				printf("num1_p= %d ₩n", *num1_p);
				printf("num2_p= %d ₩n", *num2_p);
				printf("num3_p= %d ₩n", *num3_p);

		printf("₩n");
		}
		return 0;

}
```

pdf p294 문제

q1

```
#include<stdio.h>

void cal(int *a)
{
		int i;
		for(i=0;i<7;i++){
			a[i]=a[i]*104*3/100;
			printf("%d 번 통장의 금액: %d₩n",i+1,a[i]);
		}
}


int main(void)
{
		int i =0;
		int a[7]={0};

		for(i=0;i<7;i++){
			printf("통장%d의 500만원 이하로 입금할 돈을 넣으시오₩n",i);
			printf("단위는 백만원 단위입니다.₩n");
			scanf("%d",&a[i]);
		}
		cal(a);
		return 0;

}
```

q2

```
#include<stdio.h>

void cal(int **a,int **b)
{
	int i,j,res[2];
	for(i=0;i<2;i++){
			for(j=0;j<2;j++){
				res[i]+=a[i][j]*b[j][i];
			}
			printf("₩n");
	}
	for(i=0;i<2;i++)
			printf("%3d",res[i]);

}

int main(void)
{
	int a[2][2]={{0},{0}};
	int b[2][2]={{0},{0}};
	int i, j,num;

	for(i=0;i<2;i++){
		for(j=0;j<2;j++){
		   printf("2*2 행렬의 1번 행렬 %d*%d 의 값을 입력하시오. ₩n",i,j);
		   scanf("%d",&a[i][j]);
		}
	}

	for(i=0;i<2;i++){
		for(j=0;j<2;j++){
		   printf("2*2 행렬의 1번 행렬 %d*%d 의 값을 입력하시오. ₩n",i,j);
		   scanf("%d",&b[i][j]);
		}
	}
	printf("1번 행렬은₩n");
	for(i=0;i<2;i++){
		for(j=0;j<2;j++){
			printf("%4d₩n",a[i][j]);
		}
		printf("₩n");
	}

	printf("2번 행렬은₩n");

	for(i=0;i<2;i++){
		for(j=0;j<2;j++){
			printf("%4d₩n",b[i][j]);
		}
		printf("₩n");
	}

	return 0;

}
```

삼각형 문제 Q1

```
#include<stdio.h>

int main(void)
{
	int a, b;

	printf("삼각형의 밑 변을 입력하시오.₩n");
	scanf("%d",&a);

	printf("삼각형의 높이를 입력하시오.₩n");
	scanf("%d",&b);

	printf("삼각형의 넓이 = %d₩n",a*b/2);
	return 0;
}
```

삼각형 문제 Q2

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

int angle(int a, int b)
{
	if(b<90 && b>0)
		return;
	else if(b == 90)
		return a*b/2;
	else if(b>90 && b<180)
		return ;
	else
		return 0;
}


int main(void)
{
	int a, b,c;

	printf("삼각형의 밑 변을 입력하시오.₩n");
	scanf("%d",&a);

	printf("삼각형의 밑 변과 이루는 다른변이 이루는 각도는?₩n");
	scanf("%d",&b);


	printf("삼각형의 넓이 = %d₩n",angle);
	return 0;
}
```