# TI DSP,MCU 및 Xilinux Zynq FPGA

# 프로그래밍 전문가 과정

| 이름 | 문지희 |
| --- | --- |
| 학생 이메일 | mjh8127@naver.com |
| 날짜 | 2018/3/9 |
| 수업일수 | 12 일차 |
| 담당강사 | Innova Lee(이상훈) |
| 강사 이메일 | gcccompil3r@gmail.com |

# 목차

# 1.사전평가 문제

- 단 한 번의 연산으로 대소문자 전환을 할 수 있는 연산에 대해 기술하시오.

~답

```
#include<stdio.h>

int main(void)
{
	int i;
	char arr[]="HELLOhello";
	int size=sizeof(arr)/sizeof(char)-1;
	printf("%s₩n",arr);
	for(i=0;i<=size;i++)
	{
		if(97<=arr[i])
			arr[i]=arr[i]-32;
		else if(65<=arr[i]<91)
			arr[i]=arr[i]+32;
	}
	printf("%s",arr);
}
```

-int p[7] 와 int (*p)[7] 가 있다. 이 둘의 차이점에 대해 기술하시오.

Int p[7]은 int 형 데이터를 7 개 저장할 수 있는 배열 p 이고,

Int(*p)[7]은 int 형 데이터를 7 개 저장할 수 있는 배열을 가리키는 포인터이다.

-이번 문제의 힌트를 제공하자면 함수 포인터와 관련된 문제다.
아래와 같은 행렬을 생각해보자!
1 2 3
1 2 3
sapply(arr, func) 이라는 함수를 만들어서 위의 행렬을 아래와 같이 바꿔보자!
sapply 에 func 함수가 연산을 수행하는 함수로 만들어야 한다.
1 2 3
1 4 9

~답

```
#include<stdio.h>

int func(int n)
{
	return n*n;
}
void sapply(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}};
	sapply(arr,func);
	return 0;
}
```

-이것이 없으면 C 언어의 함수를 호출할 수 없다. 여기서 이야기하는 이것은 무엇일까?

Stack?

-void (* signal(int signum, void (* handler)(int)))(int)라는 signal 함수의 프로토타입을 기술하시오.
프로토타입을 기술하라는 의미는 반환형(리턴 타입)과 함수의 이름, 그리고 인자(파라메터)가 무엇인지 기술하라는 뜻임.

리턴 : void (*)(int)

함수이름 : signal

인자 : int signum, void (* handler)(int)

-기본 for 문 활용 문제다. 1~100 까지의 숫자 중 홀수만 더해서 출력해보시오.

```
#include<stdio.h>
int odd(int s, int e){
	int sum=0;
	for(;s<=e;s++)
	{
		if(s%2==1)
			sum+=s;
	}
	return sum;
}
int main(void){
	printf("%d", odd(1,100));
	return 0;
}
```

-기본 배열 문제이다.

1 ~ 100 까지 숫자를 모두 더해서 첫 번째 배열에 저장하고
1 ~ 100 까지 숫자중 홀수만 더해서 두 번째 배열에 저장하고
1 ~ 100 까지 숫자중 짝수만 더해서 세 번째 배열에 저장한다.
다음으로 1 ~ 100 까지 숫자중 3 의 배수만 더해서 네 번째 배열에 저장한다.
각 배열의 원소를 모두 더해서 결과값을 출력하시오.

```
#include<stdio.h>
int a1(int s,int e)
{
	for(s;s<=e;s++)
		s+=s;
	return s;
}
int a2(int s,int e)
{
	for(s;s<=e;s++)
	{
		if(s%2==1)
```

```
                s+=s;
        }
        return s;
}
int a3(int s,int e)
{
        for(s;s<=e;s++)
        {
                if(s%2==0)
                        s+=s;

        }
        return s;
}
int a4(int s,int e)
{
        for(s;s<=e;s++)
        {
                if(s%3==0)
                        s+=s;
        }
        return s;
}
int main(void)
{
        int s=1,e=100;
        int arr[4];
        arr[0]=a1(s,e);
        arr[1]=a2(s,e);
        arr[2]=a3(s,e);
        arr[3]=a4(s,e);
        int sum=arr[0]+arr[1]+arr[2]+arr[3];
        printf("%d",sum);
        return 0;

}
```

- C 언어에서 중요시하는 메모리 구조에 대해 기술하시오.
(힌트: Stack, Heap, Data, Text 에 대해 기술하시오.)

Stack 은 지역변수가 저장되는 메모리공간, Heap 은 동적변수가 할당되는 메모리공간, data 는 초기화 된 전역변수와 정적변수가 저장된 메모리 공간이고 텍스트는 코드 세그먼트로 기계어로 변환된 코드가 저장 되어있다. 쓰기가 금지 되어있고 크기가 고정 되어있다.

- goto 를 사용하는 이유에 대해 기술하시오.

Goto 는 중첩된 반복문을 빠져나오기에 용이하다. 다른 방법으로 빠져나올 때 보다 코드를 간략하게 하여 빠져나올 수 있으며, 다른 경우보다 파이프라인이 적게 깨져 성능면에서 더 효과적이어서 사용한다.

-리눅스 실행 파일 포맷이 무엇인지 적으시오.

./a.out (gcc 할 때 이름설정 없을 때)

./[실행파일 이름].out

# 2.모의문제 1 회차

-변수의 정확한 정의를 기술하시오.
변수 : 데이터를 저장하는 메모리공간.

-실수와 정수를 입력받고 그 값을 함수의 인자로 넘겨 두 수의 곱을 실수형으로 반환하여 출력하시오.

```
#include<stdio.h>

float mul(int a, float b)
{
return a*b;
}

int main(void)
{
int a=4;
float b=2.4;

printf("%f",mul(a,b));
return 0;
}
```

-goto 문을 어떤 경우에 사용하면 효율적인지 작성하시오.
반복문을 빠져나올 때 코드를 간략하게 사용하여 빠져나올 수 있고 다른 방법으로 반복문을 빠져나올 때 보다 파이프라인이 적게 깨져 성능면에서도 효율적이다.

-C 언어에서 중요시 여기는 Memory 구조에 대해 기술하시오.
Stack 은 지역변수가 저장되는 메모리공간, Heap 은 동적변수가 할당되는 메모리공간, data 는 초기화 된 전역변수와 정적변수가 저장된 메모리 공간이고 텍스트는 코드 세그먼트로 기계어로 변환된 코드가 저장 되어있다. 쓰기가 금지 되어있고 크기가 고정 되어있다.

-다음 문제의 빈칸을 채우시오.
C 에서 함수를 호출하고 사용하기 위해서는 반드시 ()이 필요하기 때문에
재귀호출을 할 경우에 계속해서 ()을 생성한다.
스택

-1 ~ 1000 사이에서 짝수와 홀수의 합을 각각 구하고
짝수의 합으로 홀수의 합을 나눈 결과를 정수형으로 출력하라.

```
#include<stdio.h>
int even(int s,int e)
{
	int sum=0;
	for(;s<=e;s++)
	{
		if(s%2==0)
			sum+=s;
	}
	return sum;
}
int odd(int s, int e)
{
```

```
        int sum=0;
        for(;s<=e;s++)
        {
                if(s%2==1)
                        sum+=s;
        }
        return sum;
}

int main(void)
{
        printf("%d",even(1,1000)/odd(1,1000));
        return 0;
}
```

-7 과 4 이라는 2 개의 정수를 입력받아 num1, num2 에 저장하였다.
num1 << num2, num1 >> num2 의 값과
num1 과 num2 의 and, or, xor bit 연산을 수행하는 프로그램을
작성하시오.

```
#include<stdio.h>

int and(int num1,int num2)
{
        return num1&num2;
}
int or(int num1,int num2)
{
        return num1 | num2;
}
int xor(int num1,int num2)
{
        return num1^num2;
}
int shift1(int num1,int num2)
{
        return num1<<num2;
}
int shift2(int num1,int num2)
{
        return num1>>num2;
}
int main(void)
{
        int num1=7,num2=4;
        and(num1,num2);
        or(num1,num2);
        xor(num1,num2);
        shift1(num1,num2);
        shift2(num1,num2);
        return 0;
}
```

-Debugging(디버깅)을 왜 해야하는지 이유를 기술하고
Linux 환경에서 terminal 에 어떤 명령어를 입력하여 컴파일하는지
기술하시오.

디버깅은 컴파일은 성공적으로 문법적오류는 없지만 논리적 오류가
존재하는 경우 수행한다.
gcc -g 디버깅파일이름 c 파일 이름.c
위의 명령어를 사용하면 컴파일을 하고 중간에 작성된 파일 이름으로
디버깅 파일을 생성한다

10-11 번 복합문제
-do while 문을 사용하는 이유에 대해 기술하시오.
do while 은 한번 무조건 실행하고 while 문을 반복하고 싶을 때 반복한다.

do
{
}while(조건문)
의 형태로 소스코드를 작성함

-표준 상태에서 아래의 소스 코드가 컴파일이 되는지 안되는지 판정하시오.
만약 컴파일이 안된다면 어떻게 바꿔야 하는지 기술하시오.

```
#define inc_each(x, y) { x++; y++; }

int main(void)
{
int x = 10, y = 5;

if(x > y)
inc_each(x, y);
else
x = y;

return 0;
}
```

#define 을 없애고

```
If(x>y)
{
x++;
y++;
}
```

을 채워넣는다.

-다음 빈칸을 채우시오.
컴퓨터 구조상 가장 빠른 것은 ()이고 가장 느린것은()이다.
레지스터/ 디스크

-이중 for 문을 사용하여 구구단을 작성해보시오
.

```
#include<stdio.h>
void mul(void){
	int i,j;
	for(i=0;i<10;i++)
	{
		for(j=0;j<10;j++)
			printf("%d * %d = %d₩n",i,j,i*j);
	}
}

int main(void)
{
	mul();
	return 0;
}
```

-아래의 소스 코드를 보고 물음에 답하시오.
이 코드에 오류가 존재하는지 않는지를 판별하고
문제가 있다면 어디를 어떻게 수정하는 것이 좋을지 기술하시오.

```
#include <stdio.h>

int main(void)
{
int number = 1;
for(;;)
{
printf("number = %d₩n", number);
number += number;
if(number == 1000)
break;
}
return 0;
}
```

빨간색 부분을 if(number>=1000)으로 바꾼다.

-다음 빈칸을 채우시오.
배열의 이름은 ()이고, 포인터는 ()을 저장하는 ()이다.
대표주소/ 주소/ 변수

-구조체를 사용하는 이유에 대하여 기술하라.
여러 자료형의 묶음을 하나의 자료형으로 사용하기 위해.

24-25 복합 문제

-어떤 정수에 값 13 이 들어 있다.
이것을 4 중 포인터를 사용하여 표현해보라.

-4 중 포인터를 사용하여 표현했다면
현재 값인 13 을 14 로 증가시켜보자.
(num++을 사용하지 말라)

```
#include<stdio.h>

int main(void)
{
int i=13;
int *p1=&i;
int **p2=&p1;
int ***p3=&p2;
int ****p4=&p3;

****p4=14;
printf("%d",i);
return 0;
}
```

-전역 변수와 지역 변수의 차이점에 대해 기술하시오.
지역변수는 해당 scope 를 벗어나면 그 변수를 사용할 수 없지만
전역변수는 해당 c 파일 내에서는 어디든 사용 가능하다.

-아래 코드를 보고 화면에 출력될 결과를 예측하시오.

```
#include <stdio.h>

int main(void)
{
int shortcut1 = 0, shortcut2 = 3, num = 9;

if(shortcut1 && num++)
shortcut1++;

printf("%d, %d₩n", shortcut1, num)
```

```
if(shortcut1 || shortcut2)
shortcut1++;

printf("%d₩n", shortcut1);

if(shortcut1 && ++num)
shortcut2--;

printf("%d, %d, %d₩n", shortcut1, num, shortcut2);

return 0;
}
```

~결과
0,9
1
1,10,2

-아래 코드를 보고 화면에 출력될 결과를 예측하시오.

```
#include <stdio.h>

int main(void)
{
int i, j;
int num1, num2, num3;
int arr[2][6];
int *arr_ptr[3] = {&num1, &num2, &num3};
int (*p)[6] = arr;

for(i = 0; i < 3; i++)
{
*arr_ptr[i] = i;
printf("%d₩n", *arr_ptr[i]);
}
for(i = 0; i < 2; i++)
for(j = 0; j < 6; j++)
arr[i][j] = (i + 1) * (j + 1) * 10;

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

return 0;
}
```

~결과
0
1
2
10
20

-아래 코드를 보고 해당 프로그램이 어떻게 동작하는지 기술하시오.

```
#include <stdio.h>

int fib(int num)
{
if(num == 1 || num == 2)
return 1;
else
return fib(num - 1) + fib(num - 2);
}
```

```
int main(void)
{
int result, final_val = 6;
result = fib(final_val);
printf("%d 번째 항의 수는 = %d₩n", final_val, result);
return 0;
}
```

fib6 이 실행되고 fib(1)이나 fib(2)가 될 때 까지 fib 함수를 반복해야 한다. return 으로 fib(n-1)+fib(n-2)가 있는데 fib(5)와 fib(4)를 실행하고 각각 또 fib(4)와 fib(3), fib(3)과 fib(2)를 반복한다. 이를 반복하다보면 fib(1)이나 fib(2)가 되어 1 을 반환하는데 이 값들을 더하면 fib(6)은 8 이 출력되게 된다.

-포인터의 크기가 무엇에 따라 좌우되는지 기술하고 이유를 서술하시오.
포인터는 컴퓨터의 HW 에 따라 크기가 달라진다 64bit 시스템이면 포인터는 8byte, 32bit 시스템이면 포인터의 크기는 4byte 가 된다. 포인터는 주소를 저장하기 때문에 주소는 컴퓨터가 저장할 수 있는 최대 값을 할당해준다.

-int p[4]와 int (*p)[4]의 차이에 대해 기술하시오.
Int p[4]는 int 형 데이터를 4 개 저장 가능한 배열 p 이고,
Int(*p)[4]는 int 형 테이터 4 개를 저장하는 배열을 가리키는 포인터이다.

-함수 포인터를 사용하는 이유에 대해 기술하시오.
함수를 인자로 사용할 때 함수 포인터를 이용하여 사용한다.

-아래의 선언을 보고 이것이 무엇인지 기술하시오.
(인자는 무엇이고 반환형은 무엇인지 함수인지 함수 포인터인지 등등)

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

리턴: void (*)(int
함수이름 : signa
인자 : (int signum, void (*handler)(int))