TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 : Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 : 황수정
sue100012@naver.com
14일차 (2018. 03. 13)

11. 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;
}
```

15. 현재 디버깅중이며 메모리 주소 0x7ffffb70에 어떤 값이 저장되었다. 이 때 위 메모리에 저장된 값을 보고 싶다면 어떻게 해야 하는가 ?

16. 다음 어셈블리어를 보고 함수의 main함수부터 my_function 함수까지 stack의 동작 방식을 그림과 함께 기술하라.

```
int my_function(int num1, int num2, int num3)
{
return num2 * num3 - num1;
}

int main(void)
{
int result = 0;
int num1 = 2, num2 = 3, num3 = 4;
result = my_function(num1, num2, num3);
printf("result = %d₩n", result);
return 0;   }
```

20-22번 복합 문제

20. 20개의 배열에 rand()를 이용하여 무작위로 1 ~ 100까지의 값을 저장하는 프로그램을 작성하라.

21. 그리고 각 배열의 요소의 홀수들과 짝수들의 합을 구해 출력하라.

22. 홀수들의 합과 짝수들의 합을 곱하여 출력하라.

```
#include< stdio.h>
#include< stdlib.h> //난수 생성을 위해서 포함. srand(), rand()
#include< time.h> //난수 생성을 위해서 포함. time()

int main (void) {
   int arr[2] = {0}; // 배열을 모두 0으로 초기화시키는 선언
   int i, sum1 = 0, sum2 = 0;

for(i=0;i<20;i++)
{   arr[i] = (rand() % 100 ) + 1;
/* rand() % n : 0 ~ n-1 범위의 난수 생성
(rand() % n) +1 : 1~n 범위의 난수 생성 */
if(arr[i] % 2 == 0)
sum1 += arr[i];
else
sum2 += arr[i];   }

printf("홀수의 합 : %d,₩n 짝수의 합 : %d₩n", sum1, sum2);
printf("두 합의 곱은 %d 이다.", sum1 * sum2);
return 0;
}
```

24-25 복합 문제

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

25. 4중 포인터를 사용하여 표현했다면 현재 값인 13을 14로 증가시켜보자. (num++을 사용하지 말라)

```
#include< stdio.h>

int main (void){
 int num = 13;
```

```
    int *p = &num;
    int **pp = &p;
    int ***ppp = &pp;
    int ****pppp = &ppp;

printf("4중 포인터로 표현한 num : %d₩n", ****pppp);
/*                  ****pppp = 14;
printf("4중 포인터를 이용해서 num 변경값 %d₩n", num); */
****pppp += 1;

printf("4중 포인터를 이용해서 num 변경값 %d", num);
return 0;
}
```

26. 행렬의 곱셈을 프로그램으로 작성하라.

28. 정수형 변수 2개를 선언하고 서로 다른 값으로 초기화한다. 이후에 포인터를 사용하여 2개의 값을 서로 교환해보라.

30. 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;
}
```

31. 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;
```

```
}
```

32. 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("%d번째 항의 수는 = %d₩n", final_val, result);
return 0;
}
```

33. 임의의 값 x가 있는데, 이를 4096 단위로 정렬하고 싶다면 어떻게 해야 할까? (힌트 : 4096 = 2^12)

34. 구조체를 사용하여 Stack을 구현해보시오.

39. 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)
이름 : signal
인자 : (int signum, void (*handler)(int)) int형을 인자로 받는 함수포인터다.

40. volatile 키워드의 용도는 무엇일까 ?
>volatile로 선언된 변수는 외부적인 요인으로 그 값이 언제든지 바뀔 수 있음을 뜻한다. 따라서 컴파일러는 volatile 선언된 변수에 대해서는 최적화를 수행하지 않는다. volatile 변수를 참조할 경우 레지스터에 로드된 값을 사용하지 않고 매번 메모리를 참조한다. 메모리 주소에 연결된 하드웨어 레지스터에 값을 쓰는 프로그램이라면 이야기가 달라진다. 각각의 쓰기가 하드웨어에 특정 명령을 전달하는 것이므로, 주소가 같다는 이유만으로 중복되는 쓰기 명령을 없애버리면 하드웨어가 오동작하게 될 것이다. 이런 경우 유용하게 사용할 수 있는 키워드이다. 즉, 외부 요인에 의해 변수 값이 변경될 가능성이 있는 변수에는 volatile을 써주어야 한다.

모의고사 2

1. 아래 Code를 작성하고 이 Code의 기계어에 대한 그림을 그리고 분석하시오.

```
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;
}
```

배점 : 20점

2. RB Tree에 3, 7, 10, 12, 14, 15, 16, 17, 19, 20을 넣을 때의 그림을 그리시오.

배점 : 2점 모의고사 1회 23번 동일

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

> c언어 함수에서 **strupr()** 함수는 문자열 속의 모든 알파벳 소문자를 모두 대문자로 만드는 함수이다. 즉, 문자열 전체를 대문자화시킨다. 알파벳 외의 다른 글자 (구두점/숫자/한글/한자 등) 는 변환하지 않고 그대로 둔다. 반대로, **strlwr()** 함수는 소문자화시킨다. 두 함수 모두, 인수로 입력받은 문자열 자체를 변환한다.
string.h 헤더 파일을 인클루드해주어야 사용할 수 있다.
> 문자열이 아닌 문자(char) 1개의 대소문자를 변환하는 방법 또한 있다. #inculde <ctype.h> 선언해주고
toupper() 함수를 쓰면 소문자를 대문자로 바꿔준다. **tolower()** 함수는 대문자를 소문자로 바꿔준다.
두 함수 모두 char 가 아닌 int형으로 입출력한다. char는 int 속에 포함되므로, char나 int 나 둘 다 사용 가능하다. (int형으로도 글자를 표현할 때는 아스키 코드 값으로 사용한다.)
> 연산으로 이용하는 경우 아스키코드를 사용하면 된다. +-32를 해주면 되는데 이는 아스키코드에서 대소문자가 32만큼 차이 나기 때문이다.

```
#include <stdio.h>

int main (void) {
    char a = 0;

    printf("문자를 입력해주세요 : ");
    scanf("%c",& a);

    if (a >= 97 && a <= 122) // 아스키코드에서 97~122 사이가 소문자
        printf("%c₩n", a - 32); // -32하면 대문자 범위로 간다.
    else if ( a >= 65 && a <= 90)
        printf("%c₩n", a + 32);
    else
        printf("잘못 입력하셨습니다.₩n");

return 0;
}
```

배점 : 2점
4. Stack과 Queue의 차이점에 대해 작성하시오.
스택은 **LIFO(last in first out)**이고, 큐는 **FIFO(first in first out)**이다.

스택은 자료구조에서는 **무언가를 쌓는다라는 의미**를 갖는 자료구조이다. 즉, 자료를 순서대로 쌓아서 보관하고 사용한다. 밑이 막힌 통에 값을 차례대로 저장해놨다가 그 값이 필요 할 때는 가장 위에서부터 꺼내는 구조라고 생각하면 된다. 예를 들면, C에서 함수를 호출할 때, 현재 함수에서 사용되는 값을 스택에 집어넣고 부르는 함수에 쓰일 값들에 대한 작업을 시작한다. 작업이 끝나면, 함수를 리턴하고 스택에 넣었던 값을 꺼내는 것이다. 이렇게 하면 함수가 호출되기 이전의 상태로 복원 시킬 수 있다.
큐는 스택과 반대로 뻥 뚫린 통에 값을 차례대로 저장해놓으면 넣은 순서대로 값이 빠져나오는 구조라고 생각하면 된다.

배점 : 3점

5. 임의의 값 x가 있는데, 이를 8192 단위로 정렬하고 싶다면 어떻게 해야 할까? (힌트 : 8192 = 2^13)

배점 : 2점

6. int *p[4]와 int (*p)[4]의 차이에 대해 기술하시오.

int *p[4] 주소를 저장 가능한 배열 p로 포인터 배열이다.
int (*p)[4] = int(*)[4] p 로 배열 포인터이고 int 형 데이터 4개를 저장 가능한 배열을 가르키는 포인터이다.

배점 : 2점

7. int[4][4]배열이 메모리 공간에 나란히 존재함을 증명하라!

```
#include <stdio.h>

int main (void) {
   int arr[4][4]={0}; //배열을 모두 0으로 초기화시키는 선언
   int i, j;

for(i = 0; i<4; i++)
{
for(j = 0; j<4; j++)
printf("arr[%d][%d]의 주소는 %p₩n", i, j, &arr[i][j]);   }

return 0;
}
```

배점 : 2점

8. int num = 7, 육중 포인터(******)를 이용해서 이를 표현하라!

```
#include <stdio.h>

int main (void) {
int num = 7;

int *p = &num;
int **pp = &(p);
int ***ppp = &(pp);
int ****pppp =& (ppp);
int *****ppppp =& (pppp);
int ******pppppp =& (ppppp);

printf("6중 포인터로 표현한 p : %d₩n", ******pppppp);

return 0;   }
```

>()가 생긴 이유는 처음에 깜박하고 int **pp = &*p; 라고 입력을 했었다. 그리고 경고가 떴었고, 혹시 제대로 못 읽은 건가 하고 ()를 삽입했다. 그러나 &는 주소를 읽는, 받아오도록 하는 것이고 *은 주소에 접근하는 것으로 둘을 함께 쓰면 상쇄된다. 따라서 값을 읽을 수가 없기 때문에 지금 작성한 답이 맞는 것이다.

배점 : 3점 / 모의고사 1의 35번과 동일

9. char *str = "Pointer Is Very Important"라는 문자열이 있다. 여기에 대문자가 총 몇 개 사용되었는지 세어보자(프로그래밍으로)

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

int main (void)   {
   char *str = "Pointer Is Very Important";
   int i;
   int len = strlen(str); // 길이를 확인해서 조건식을 걸기 위함
   int count = 0;
```

```
    for(i = 0; i <= len; i++)
      { if(str[i] >= 65 && str[i] <=90)
          { count ++; // 후위 연산자로 조건에 맞을 경우 카운트가 된다.
            printf("%c₩n", str[i]);//사용된 대문자의 갯수가 맞게 출력되었나 확인하기
                                    위해 사용함.
          }
       }
printf("대문자는 총 %d개 사용되었다.₩n", count);
return 0;   }
```

배점 : 3점

10. int arr1[3][3] = {{1, 3, 6}, {7, 1, 9}, {3, 7, 5}}과 int arr2[3][3] = {{3, 5, 7}, {9, 1, 8}, {5, 1, 2}}중 배열의 각 요소의 합이 가장 큰 것은 무엇인가 ?(프로그래밍)

```
#include <stdio.h>

int main (void){
   int arr1[3][3] = {{1, 2, 3}, {7, 1, 9}, {3, 7, 5}};
   int arr2[3][3] = {{3, 5, 7}, {9, 1, 8}, {5, 1, 2}};
   int i, j, sum1, sum2;

for(i = 0; i < 3; i++)
{
for(j = 0; j < 3; j++)
{      sum1 += arr1[i][j];
         sum2 += arr2[i][j]; }
}

printf("arr1의 합은 %d이고 arr2의 합은 %d이다.₩n arr1 < arr2 일 때, 값은 %d이다
.₩n 이 값은 참이면 1 거짓이면 0이다.₩n", sum1, sum2, sum1 < sum2);

return 0; }
```

>arr1의 합은 1942520886이고 arr2의 합은 32754이다. arr1 < arr2 일 때, 값은 0이다.
이 값은 참이면 1 거짓이면 0이다.
이므로 arr2 가 더 크다.

배점 : 1점

11. int형 변수 3개를 할당하고자 하여 아래와 같은 코드를 작성하였다. int *dyn_arr = (char *)malloc(sizeof(char) * 3);에서 잘못된 부분을 정정하라

> int ***dyn_arr = (int***)malloc(sizeof(int)*3);

배점 : 2점

12. 아무 문자열을 calloc을 통해 동적할당하고 그 문자열을 배열로 옮겨보라

배점 : 3점 모의고사 1 29번과 일치

13. 임의의 구조체를 한 개 만들고 해당 구조체에 대한 변수 2개를 선언한 이후 구조체의 멤버들에 적절한 값을 입력한 후 해당 값을 서로 교환해보라

```
#include <stdio.h>

typedef struct{
  int a;
  int b;
} A;

int main (void) {
   A d1 = {4, 5};
   A d2 = {8, 9};

printf("A d1 = %d, %d를 가지고, ₩n", d1.a, d1.b);
printf("A d2 = %d, %d를 가진다.₩n", d2.a, d2.b);

int v = 0;

v = d1.a;
d1.a = d2.a;
d2.a = v;
v = d1.b;
d1.b = d2.b;
d2.b = v;

printf("변환 후 A d1 = %d, %d를 가지고, ₩n", d1.a, d1.b);
```

```
printf("A d2 = %d, %d를 가진다.₩n", d2.a, d2.b);

return 0;
}
```

배점 : 3점

14. 7명의 직원에 대한 급여를 입력받는다. 이들이 받는 급여의 평균을 출력하고 평균 이상을 받는 사람들의 이름을 출력하라.

```
#include <stdio.h>

typedef struct {
        char name[10];
        int pay;
}employee;
int main(void){
        employee p[7];
        int sum = 0;
        double ave = 0;
        int i;
        for (i = 0; i < 7; i++)
        {       printf("이름을 입력하고 급여를 입력하시오.");
                scanf("%s %d", p[i].name, &p[i].pay);
                sum += p[i].pay;        }
        ave = sum / 7;
        printf("평균 급여 = %f₩n", ave);
        printf("평균 이상 급여자₩n");
        for (i = 0; i < 7; i++)
        {               if (p[i].pay > ave)
                        printf("%s %d₩n", p[i].name, p[i].pay);   }
        return 0;
}
```

배점 : 2점

15. Intel Architecture와 ARM Architecture의 차이점은 ?

우선 모든 프로세서는 레지스터에서 레지스터로 연산이 가능하다. 인텔 x86 은 메모리에서 메모리로 연산이 가능하다. 하지만 ARM 은 로드/스토어 아키텍처라고 해서 메모리에서 메모리로 연산이 불가능하다. 반도체 다이 사이즈가 작기 때문이다. 다이 사이즈가 작아서 Functional Unit 갯수가 적다. 이런 연산을 지원해줄 장치가 적어서 안 된다. 그래서 ARM 은 먼저 메모리에서 레지스터로 값을 옮기고 다시 이 레지스터 값을 메모리로 옮기는 작업을 한다. 로드하고 스토어하는 방식이라고 해서 로드/스토어 아키텍처라고 한다.

배점 : 2점

16. 이것이 없으면 C 언어를 사용할 수 없다. 이것은 무엇일까 ?

>헤더파일?? 컴파일러???? 스택??? 메모리 주소??? 메인함수??????

배점 : 2점

17. 3을 좌측으로 3 Shift 하는 코드를 작성하고 8을 우측으로 2 Shift 하는 코드를 작성하시오.

```
#include <stdio.h>

int Rshift(int num, int a)
{       return num >> a; }
int Lshift(int num, int a)
{       return num << a; }

int main(void){
        printf("%d₩n", Lshift(3, 3));
        printf("%d₩n", Rshift(8, 2));

        return 0;
}
```

배점 : 4점

18. 1, 3, 4, 7, 11, 18, 29, 47, 76, ... 형태로 숫자가 진행된다. 40번째 숫자는 무엇일까 ?

배점 : 4점

19. 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ... 형태로 숫자가 진행된다. 1 ~ 30번째까지의 수들로 홀수들의 합을 하고 짝수들의 합을 구한다. 홀수들의 합 - 짝수들의 합의 결과를 출력하

시오.

배점 : 2점

20. 3 by 3 행렬의 덧셈을 계산하는 프로그램을 작성하시오.

[복합문제]

배점 : 3점

21. 난수를 활용해서 Stack을 구성한다. (같은 숫자가 들어가지 않게 하고 20개 이상 넣는다. 이때 들어가는 숫자는 1 ~ 50사이의 숫자로 넣는다)

배점 : 5점

22. 그 중에서 홀 수만 빼내서 AVL Tree를 구성하도록 한다.

배점 : 5점

23. 나머지(짝 수)는 Binary Tree로 구성한다.

배점 : 1점

24. Linux에서 vim Editor는 굉장히 사랑받는 Editor 중 하나다. 여기서 7줄 복사해서 붙여넣고자 하는데 어떻게 해야 7줄을 복사 붙여넣기 할 수 있는가 ?

명령어 모드에서 y7y 로 복사하고자 하는 7줄을 복사하고 붙여 넣고자 하는 자리에 p로 붙여 넣기 한다.

배점 : 2점

25. 프로그램을 최적화하여 컴파일 하는 옵션을 적고 최적화하지 않고 디버깅 옵션을 넣어서 컴파일 하는 방법을 기술하시오.

리눅스에서

최적화하여 하는 방법은 gcc 파일명 으로 하면 된다. 컴파일이 진행되어 실행 파일이 생성된다.

gcc -g -o0 파일명 으로 입력하여 한다. 여기서 , '-g'는 디버깅 옵션이고 '-o0' 이란 명령어가 모든 최적화를 방지하는 명령어이다.