TI DSP,Xilinx zynq FPGA,MCU 및 Xilinx zynq FPGA 프로그래밍 전문가 과정

강사-INNOVA LEE(이상훈)

Gccompil3r@gmail.com

학생-윤지완

Yoonjw7894@naver.com

오답노트

2. 배열에 아래와 같은 정보들이 들어있다.

2400, 2400, 2400, 2400, 2400, 2400, 2400,

1, 2, 3, 4, 5, 1, 2, 3, 4, 5,

2400, 2400, 2400, 2400, 2400, 2400, 1, 2, 3, 4,

5, 1, 2, 3, 4, 5, 2400, 2400, 2400, 2400, 2400, 5000,

1, 2, 3, 4, 5, 5000, 5000, 500, 500, 500, 500, 500,

1, 2, 3, 4, 5, 500, 500, 500, 500, 500, 500, 500,

1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5,

1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5,

1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5,

1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5,

1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5,

1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5,

1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,

11, 12, 13, 14, 15, 16, 17, 18, 234, 345, 26023, 346, 345, 234,

457, 3, 1224, 34, 646, 732, 5, 4467, 45, 623, 4, 356, 45, 6, 123,

3245, 6567, 234, 567, 6789, 123, 2334, 345, 4576, 678, 789, 1000,

2400, 2400, 2400, 2400, 2400, 2400, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5,

2400, 2400, 2400, 2400, 978, 456, 234756, 5000, 5000, 5000, 2400, 500, 5000, 2400, 500,

500, 500, 500, 500, 500, 1, 2, 3, 4, 5, 500, 500, 500, 500, 500,

500, 500, 500, 500, 500, 1, 2, 3, 4, 5, 500, 500, 500, 5000, 2400, 5000,

5000, 5000, 5000, 5000, 5000, 5000, 5000, 1, 2, 3, 4, 5, 5000, 5000,

5000, 5000, 2400 5000, 500, 2400, 5000, 5000, 5000, 5000, 5000, 5000, 5000, 5000,

1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 5000, 5000, 5000, 5000, 5000,

1, 2, 3, 4, 5, 5000, 5000, 5000, 5000, 5000, 234, 4564, 3243, 876,

645, 534, 423, 312, 756, 235, 75678, 2400, 5000, 500, 2400, 5000, 500, 2400, 5000,

500, 2400, 5000, 500, 2400, 5000, 500, 2400, 5000, 500, 2400, 5000, 500, 2400, 5000, 500, 2400,

5000, 500, 2400, 5000, 500, 7, 8, 9, 6, 7, 8, 9, 6, 7, 8, 9, 6, 7, 8, 9, 6, 7, 8, 9, 6, 7, 8,

9, 6, 500, 2400, 5000, 500, 2400, 5000, 500, 2400, 5000, 500, 2400, 5000, 500, 2400, 5000,

500, 2400, 5000,

여기서 가장 빈도수가 높은 3 개의 숫자를 찾아 출력하시오!

함수에서 이 작업을 한 번에 찾을 수 있도록 하시오.

(찾는 작업을 여러번 분할하지 말란 뜻임)

->

```
#include<stdio.
h>

typedefstruct map
{
int freq;
int cnt;
} hash;
```

```
int dup_cnt;

hash dup[100];


intcheck_dup(int num)

{

int i;


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

if (dup[i].freq == num)

return i;


return -1;

}


intfind_kinds_freq(int *arr, int size)

{

int i;


/*

1. 값을 넣고

2. 다음에 들어오는 값이 이전 값중 같은게 있다면

3. cnt 를 증가시킨다.

*/

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

 {
```

```
int dup_loc;
if ((dup_loc = check_dup(arr[i])) != -1)
 {
if (arr[i] == 1)
printf("Here₩n");
dup[dup_loc].cnt++;
 }
else
 {
dup[dup_cnt].freq = arr[i];
dup[dup_cnt].cnt++;
 dup_cnt++;
 }
 }

return dup_cnt;
}

voidprint_arr(hash *arr)
{
int i;

for (i = 0; i < dup_cnt - 1; i++)
printf("freq = %d, cnt = %d₩n", arr[i].freq, arr[i].cnt);
```

```
puts("");
}

voidsort(hash *arr, int len)
{
int i, j, key1, key2;

for (i = 1; i < len - 1; i++)
 {
 key1 = arr[i].freq;
 key2 = arr[i].cnt;

for (j = i - 1; arr[j].freq > key2 && j > -1; j--)
 {
 arr[j + 1].freq = arr[j].freq;
 arr[j + 1].cnt = arr[j].cnt;
 }

 arr[j + 1].freq = key1;
 arr[j + 1].cnt = key2;
 }
}

voidprint_many_freq(void)
{
```

```
int i;

for(i = dup_cnt - 2; i > dup_cnt - 5; i--)
printf("dup[%d].freq = %d₩n", i, dup[i].freq);
}

intmain(void)
{
int arr[] = {
2400, 2400, 2400, 2400, 2400, 2400, 2400, 1, 2, 3, 4, 5,
1, 2, 3, 4, 5, 2400, 2400, 2400, 2400, 2400, 2400,
1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 2400, 2400, 2400, 2400, 2400,
5000, 1, 2, 3, 4, 5, 5000, 5000, 500, 500, 500, 500, 500,
1, 2, 3, 4, 5, 500, 500, 500, 500, 500, 500, 500,
1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5,
1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5,
1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5,
1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5,
1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5,
6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 234, 345,
26023, 346, 345, 234, 457, 3, 1224, 34, 646, 732, 5, 4467,
45, 623, 4, 356, 45, 6, 123, 3245, 6567, 234, 567, 6789,
123, 2334, 345, 4576, 678, 789, 1000, 2400, 2400, 2400, 2400,
2400, 2400, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 2400, 2400, 2400,
2400, 978, 456, 234756, 5000, 5000, 5000, 2400, 500, 5000,
```

```
2400, 500, 500, 500, 500, 500, 500, 1, 2, 3, 4, 5, 500, 500,

500, 500, 500, 500, 500, 500, 500, 500, 1, 2, 3, 4, 5, 500,

500, 500, 5000, 2400, 5000, 5000, 5000, 5000, 5000, 5000,
5000,

5000, 1, 2, 3, 4, 5, 5000, 5000, 5000, 5000, 2400, 5000, 500,

2400, 5000, 5000, 5000, 5000, 5000, 5000, 5000, 5000,

1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 5000, 5000, 5000, 5000, 5000,

1, 2, 3, 4, 5, 5000, 5000, 5000, 5000, 5000, 234, 4564, 3243,

876, 645, 534, 423, 312, 756, 235, 75678, 2400, 5000, 500,

2400, 5000, 500, 2400, 5000, 500, 2400, 5000, 500, 2400,
5000,

500, 2400, 5000, 500, 2400, 5000, 500, 2400, 5000, 500, 2400,

5000, 500, 2400, 5000, 500, 7, 8, 9, 6, 7, 8, 9, 6, 7, 8,

9, 6, 7, 8, 9, 6, 7, 8, 9, 6, 7, 8, 9, 6, 500, 2400, 5000,

500, 2400, 5000, 500, 2400, 5000, 500, 2400, 5000,

500, 2400, 5000, 500, 2400, 5000

 };


int size = sizeof(arr) / sizeof(int);

printf("size = %d₩n", size);


 hash *map_arr = NULL;


 size = find_kinds_freq(arr, size);

printf("dup_cnt = %u₩n", dup_cnt);

print_arr(dup);
```

```
sort(dup, size);

print_arr(dup);


print_many_freq();


return0;

}
```

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.

```
#include
<stdio.h>


        #defineRESOLUTION1 << 12

        #defineMINUS_VOLT -5

        #definePLUS_VOLT5


        floatget_slice(void)

        {
```

```
//return (float)(PLUS_VOLT - MINUS_VOLT) / (1 << 16);
return (float)(PLUS_VOLT - MINUS_VOLT) / (RESOLUTION);
}

floatadc(int bit, float slice)
{
return (float)(MINUS_VOLT) + bit * slice;
}

intmain(void)
{
float slice, volt;
int bit = 2077;

  slice = get_slice();
printf("slice = %f₩n", slice);

  volt = adc(bit, slice);
printf("volt = %f₩n", volt);

return0;
}
```

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)

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

                intwhere_is_it(char *text, char *find)
                {
                int i;

                for(i = 0; text[i]; i++)
                if(!(strncmp(&text[i], find, strlen(find))))
                return i;
                }

                intmain(void)
                {
                int idx;
                char text[100] = "TI Cortex-R5F Safety MCU is very good to Real-Time
```

```
System.";

  idx = where_is_it(text, "Safety MCU");
printf("idx = %d₩n", idx);

return0;
}
```

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

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

voidinit_mat(int (*M)[3])
{
int i, j;

for(i = 0; i < 3; i++)
for(j = 0; j < 3; j++)
  M[i][j] = rand() % 4 + 1;
}

voidprint_mat(int (*M)[3])
{
```

```
int i, j;


for(i = 0; i < 3; i++)
 {
for(j = 0; j < 3; j++)
 {
printf("M[%d][%d] = %d₩t", i, j, M[i][j]);
 }
printf("₩n");
 }
}


voidmult_mat(int (*A)[3], int (*B)[3], int (*R)[3])
{
int a = A[0][0], b = A[0][1], c = A[0][2],
 d = A[1][0], e = A[1][1], f = A[1][2],
 g = A[2][0], h = A[2][1], i = A[2][2],
 j = B[0][0], k = B[0][1], l = B[0][2],
 m = B[1][0], n = B[1][1], o = B[1][2],
 p = B[2][0], q = B[2][1], r = B[2][2];


 R[0][0] = a * j + b * m + c * p;
 R[0][1] = a * k + b * n + c * q;
 R[0][2] = a * l + b * o + c * r;
 R[1][0] = d * j + e * m + f * p;
```

```
  R[1][1] = d * k + e * n + f * q;
  R[1][2] = d * l + e * o + f * r;
  R[2][0] = g * j + h * m + i * p;
  R[2][1] = g * k + h * n + i * q;
  R[2][2] = g * l + h * o + i * r;
}

intmain(void)
{
int A[3][3] = {0};
int B[3][3] = {0};
int R[3][3] = {0};

srand(time(NULL));

init_mat(A);
init_mat(B);

printf("A:₩n");
print_mat(A);
printf("₩nB:₩n");
print_mat(B);

mult_mat(A, B, R);
```

```
printf("₩nR:₩n");
print_mat(R);

return0;
}
```

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)

```
#include<stdio.
h>

// int (*)(int)
intt1(int a)
{
printf("t1 function₩n");
return a * 3;
}

// int (*)(int) tp1(int (*p)(int))
// int (* tp1(int (*p)(int)))(int)
int (* tp1(int (*p)(int)))(int)
{
```

```
printf("tp1 function₩n");

p(3);

return p;

}


// int (*)(int) (*)(int (*p)(int)) tpp1 (int a, int
b)

// int (*)(int) (* tpp1 (int a, int b))(int
(*p)(int))

// int (* (* tpp1 (int a, int b))(int
(*p)(int)))(int)

int (* (* tpp1(int a, int b))(int (*p)(int)))(int)

{

printf("tpp1 function a = %d, b = %d₩n", a,
b);

return tp1;

}


intmain(void)

{

int num;

 num = tpp1(3, 7)(t1)(33);

printf("num = %d₩n", num);

return0;

}
```

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

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


                    intcalc_piece(float interval, float dx)
                    {
                    return interval / dx;
                    }


                    floatcalc_int_4x_2_5x_1(float start, float end)
                    {
                    int i, int_s, int_e;
                    float sum = 0.0f;
                    float temp = 0.0f;
                    constfloatexp = 2.0;
                    //const float dx = 0.000001f;
                    //const float dx = 0.0000001f;
                    //const float dx = 0.0000002f;
                    //const float dx = 0.0000004f;
                    //const float dx = 0.0000008f;
                    //const float dx = 0.0000016f;
                    //const float dx = 0.000002f;
                    //const float dx = 0.0000025f;
                    //const float dx = 0.000003f;
```

```
//const float dx = 0.0000027f;

//const float dx = 0.0000026f;

//const float dx = 0.0000024f;

//const float dx = 0.0000022f;

//const float dx = 0.00000255f;

//const float dx = 0.00000252f;

//const float dx = 0.00000253f;

//const float dx = 0.00000251f;

//const float dx = 0.000002505f;

constfloat dx = 0.000002504f;


float dy = pow(dx, exp);


printf("dx = %.3f₩n", dx);

printf("dy = %.6f₩n", dy);


 int_s = 0.0f;

 int_e = calc_piece(end - start, dx);


printf("int_e = %d₩n", int_e);


for (i = int_s; i < int_e; i++)

 {

 temp += dx;

 dy = 4 * pow(temp + start, exp) + 5 * (temp + start) +
1;
```

```
 sum += dx * dy;
 }

printf("sum = %f₩n", sum);

return sum;
}

intmain(void)
{
float res;
 res = calc_int_4x_2_5x_1(1.0f, 3.0f);
printf("res = %f₩n", res);
return0;
}
```

13. 12 번 문제에서 각 배열은 물건을 담을 수 있는 공간에 해당한다.

앞서서 100 개의 공간에 물건들을 담았는데 공간의 낭비가 있을 수 있다.

이 공간의 낭비가 얼마나 발생했는지 파악하는 프로그램을 작성하시오.

```
#include<stdio.
h>

voidmake_fib(int *p, int last)
{
int i = 2;

for(; i < last; i++)
```

```
 p[i] = p[i - 2] + p[i - 1];
}

intget_even_odd_diff(int *p)
{
int i, even = 0, odd = 0;

for(i = 0; i < 27; i++)
 {
if(p[i] % 2)
 odd += p[i];
else
 even += p[i];
 }

return odd - even;
}

intmain(void)
{
int res;
int fib_arr[27] = {1, 1};
make_fib(fib_arr, 27);
 res =
get_even_odd_diff(fib_arr);
```

```
printf("res = %d₩n", res);

return0;
}
```

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

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

voidchg_order(char *str)
{
int i, len = strlen(str);

printf("len = %d₩n", len);

for(i = 0; i < len; i++)
if((str[i] >= 97 && str[i] < 123) || (str[i] >= 65 && str[i] < 91))
  str[i] ^= 0x20;
}

intmain(void)
{
char test[10] = "Who am I";
```

```
printf("test = %s₩n", test);

chg_order(test);

printf("test = %s₩n", test);


return0;
```

.