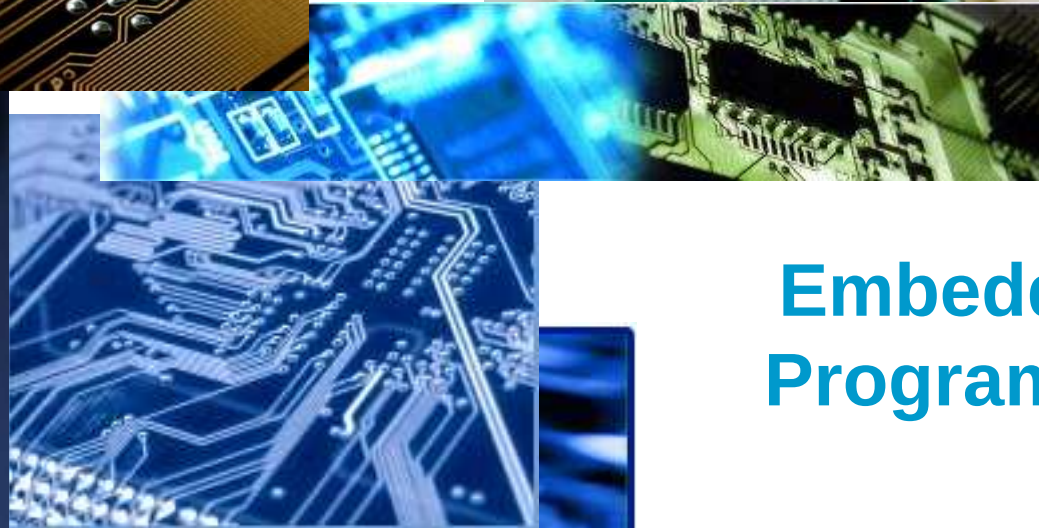
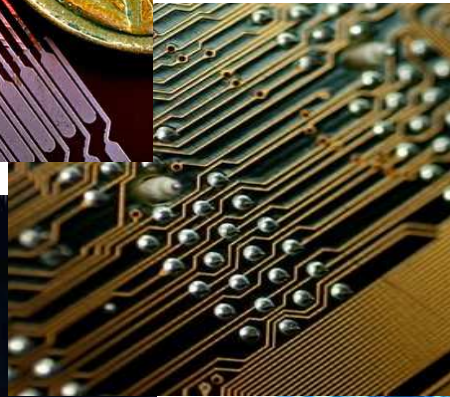
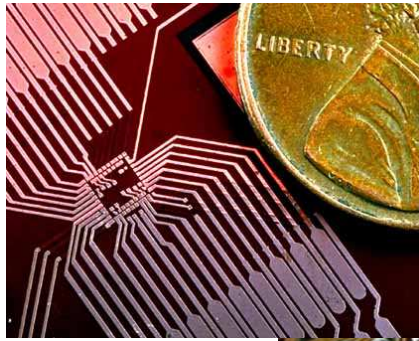




Embedded C Programming

Ramy Souody





Agenda:

- Pointers
- Open Discussion
- Interactive Questions



Agenda:



Pointers

- Open Discussion
- Interactive Questions

Pointers:

- Pointer is a variable in memory where we can store address.
- Pointer scope and lifetime are same as normal variables.
- Pointer size is not standard and it depends on the memory architecture and memory space.

Pointers:

- Pointer definition:
 - `base_type * pointer_name;`
- Example:
 - `char * ptr;`

Pointers:

- Example:

```
char x,z;
```

```
ptr = &x;
```

```
x = 5;
```

Address of x

```
*ptr = 5;
```

Value at ptr

```
z = *ptr;
```

```
z = x;
```

Pointers:

- Pointer Arithmetic:

short x;

char * pc;

pc = &x; (Error)

- Step size:

short * pc;

pc++;

(step size is according to the base type)

Pointers:

- Operators that we can use with pointers:

$+$, $-$, $++$, $--$

- Note:
 - $\text{ptr2} - \text{ptr1}$ = number of elements not bytes

Pointers:

- Example:

```
char n = 10;      // address 100
```

```
char * p;
```

```
p = &n;
```

```
(*p)++;
```

```
printf("%p",p);    // 100
```

```
printf("%d", *p);  // 11
```

Pointers:

- Pointers and arrays:

```
char st[4] = {0,1,2,3};
```

```
char * pc;
```

```
pc = st;           // pc = &st[0];
```

```
printf("%d", *(pc+2));
```

Pointers:

- Example:

```
char ch;
```

```
ch = st[1];      // array indexing
```

```
ch = pc[1];      // pointer indexing
```

```
ch = *(pc+1);    //pointer arithmetic
```

Pointers:

- Note:

```
short s[10], *ptr;
```

```
ptr = s;
```

```
ptr += 3;
```

- This statement adds 3 elements (3*step size) not 3 bytes.

Pointers:

- Pointer as a function parameter:

```
void fill_array (char Ar[], char size, char value)
{
    char i;
    for (i=0;i<size;i++) Ar[i] = value;
}

void main (void)
{
    char y[5];
    fill_array(y,5,0);
}
```

Pointers:

- Pointer as a function parameter:

```
void fill_array (char * p, char size, char value)
{
    char i;
    for (i=0;i<size;i++) p[i] = value; // we can use pointer
                                        arithmetic here
}

void main (void)
{
    char y[5];
    fill_array(y,5,0);
}
```

Pointers:

- Note:
 - Pointers can be used for more than one return value from a function.

```
void func_arth (char x, char * square, char * cube)
{
    *square = x*x;
    *cube = x*x*x;
}
```

Pointers:

- Pointer to function:
 - Definition:
`Return_type (*ptr_name) (argument_type argument_name);`
 - Initialization:
`Ptr_name = func_name;`
 - Calling:
Same as function call;

Pointers:

- Example:

```
char square, cube;
```

```
void (*ptr) (char x, char * square, char * cube);
```

```
ptr = func_arth ;
```

```
ptr(3 , &square, &cube);
```



Agenda:

- Pointers
- ➔ Open Discussion
- Interactive Questions



Open Discussion:

- Where are your Questions ? 😊



Agenda:

- Pointers
- Open Discussion
- ➔ Interactive Questions

Interactive Questions

- In the following program, how would you print 50 using p?

```
void main (void)
{
    char a[] = {10,20,30,40,50};
    char * p;
    p = a;
    printf(.....);
}
```

Interactive Questions

- In the following program, add a statement in the function fun() such that address of a is stored in j;

```
void fun (int **);  
main()  
{  
    int * j;  
    fun(&j);  
}  
void fun (int ** k)  
{  
    int a = 10;  
    /*add statement here*/  
}
```

Interactive Questions

- Would the following program give any warning on compilation?

```
main()
{
    float * p1, i = 25.5;
    char * p2;
    p1 = &i;
    p2 = &i;
}
```



info@escommittee.com

education@escommittee.com