1. I/O Port at address 0x0900. Write a byte to it. Read a byte from it.

Pretending DDRA port address is 0x0900

DDRA &= 0xff // writing one bit

Uint8\_t read\_byte = DDRA & oxff; // atmel as an example

1. 2. Block of Memory at address 0xF0000000, size (in bytes) is 0x12345. Fill it with pattern 0xAABBCCDD using for loop. Also, describe what type of problems could be incurred?

First of all should consider the storage sequence, from lower byte to higher byte, so fill it with pattern 0xAABBCCDD 4 bytes. It should be divided for four parts, 0xAA 0xBB 0xCC 0xDD and should store 0xDD first, since 0xDD is the lowest byte.

Int \* b = (int \*)0xF0000000;

For ( int i = 0; i < 0x12345/4; i++)

{

\*b++ = 0xDD; \*b++ = 0xCC;\*b++=0xBB; \*b++ = 0xAA;

}

// Int rest = 0x12345 – (int)(0x12345/4)\*4 = 1; so there is more byte left to fill

\*b = 0xDD;

Problem 1: The 4 bytes number, 0xAABBCCDD cannot be stored totally in the last several bytes, only the lowest byte 0xDD can be stored.

Problem 2: the memory size is 0x12345, this is not reasonable, should be even.

1. Define a linked list of memory blocks. Allocate a block of 256 bytes and insert into the first node of list. Allocate a 2nd block of same size and add to the list. Then free each block from the list.

#include <stdio.h>

#include<stdlib.h>

Struct block\_node{

Char mem[256];

Struct block\_node \* next;

};

Int main(void){

Struct block\_node \* head, \*Pnext;

head = (struct block\_node \*)malloc(sizeof(struct block\_node)); // this is head/first of the list with the unit of 256 bytes memory.

// inset Pnext to be the next

//allocate a space for Pnext

Pnext = (struct block\_node \*)malloc(sizeof(struct block\_node)); //

head->next = Pnext;

//the ending node

Pnext - > next = NULL;

// and free the space

Free(Pnext);

Free(head);

Return 1;

}

1. Send a message containing “Hello World” via TCP/IP socket to device located at 198.0.0.1. Then listen for a response and print the response.

Linux Example, just only my understanding :

|  |  |
| --- | --- |
| Server (IP 198.0.0.1) | Client |
| #include <sys/socket.h>  Int main()  {  int socket\_id, connect\_id;  struct sockaddr\_in servaddr;  // create a socket  int socket\_id = socket(AF\_INET,Sock\_Stream,0);  // set parameters  servaddr.sin\_family = AF\_INET;  servaddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);  \_  servaddr.sin\_port = 1234;  //bind socket id  bind(socket\_fd, (struct sockaddr\*)&servaddr, sizeof(servaddr);  //listen for the information from client  listen(socket\_fd, MAXLINK);  //accept the requirement from client  connect\_id= accept(socket\_fd, (struct sockaddr\*)NULL, NULL);  // successful connect  for(;;)  {  // server start to receive data  Char buff[100];  Recv(connect\_id,buff,100,0);  Send(connect\_fd,buff,100,0);  Cout<<buff<<endl;  }  //close id flag  Close(connect\_id);  Close(socket\_id);  } | #include <sys/socket.h>  Int main(){  Int socket\_id;  struct sockaddr\_in servaddr;  // create a socket  int socket\_id = socket(AF\_INET,Sock\_Stream,0);  // ping 198.0.0.1  servaddr.sin\_family = AF\_INET;  servaddr.sin\_addr.s\_addr = htonl(198.0.0.1);  \_PO  servaddr.sin\_port = 1234;  // start to connect  connect(socket\_id,(sockaddr\*)& servaddr,sizeof(servaddr));  // if connection is o, send message hello world  write(socket\_id,”hello world”,100,0);  // and then listening  Char buff[100];  Read(socket\_id,buffer,100);  //close the id  Close (socket\_id);  } |

5. Take a random 32-bit memory address, and align it to the nearest 1KB address boundary.

#include<stdio.h>

#include<string.h>

Int main(void)

{

// input the add

Int address;

Int \* test = (int \*) address;

// since 1 K = 1024;

//find which side is nearer

If(test%1024 < 512)

{

Test -= test%1024;

}

Else

{

Test +=2014 – test%1024;

}

Cout<<Test<<endl;

}