2022 CAT-CERT 스터디 보고서

컴퓨터정보공학부 202221168 김지혜

2022.05.13#System\_study-3

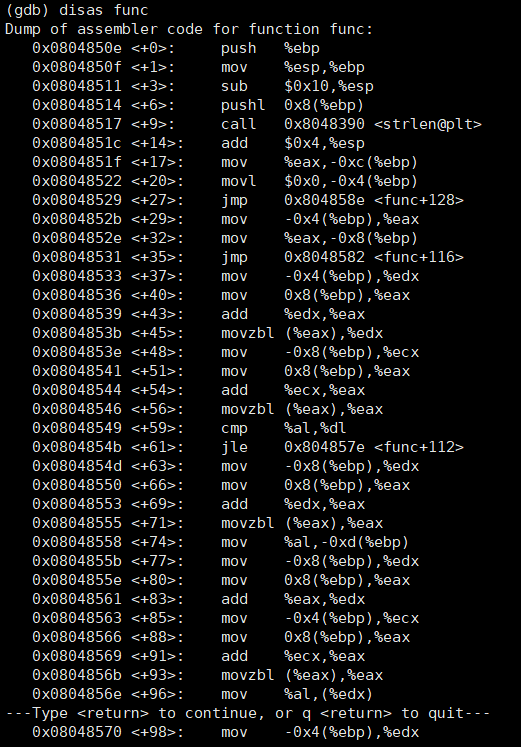
**Prob5**

**소스코드**

|  |
| --- |
| **#include<stdio.h>**  **#include<string.h>**  **int func(char\*str);**  **int main(){**  **char str[20];**    **printf("INPUT ->");**  **scanf("%s",str);**    **func(str);**  **puts(str);**  **}**  **int func(char \*str){**  **int i,j,len;**  **char temp;**  **len=strlen(str);**  **for(i=0; i<len;i++){**  **for(j=i;j<len;j++){**  **if(str[i] > str[j]){**  **temp=str[j];**  **str[j]=str[i];**  **str[i]=temp;**  **}**  **}**  **}**  **}** |

**출력결과**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**텍스트, 명판이(가) 표시된 사진

자동 생성된 설명**

**Prob7**

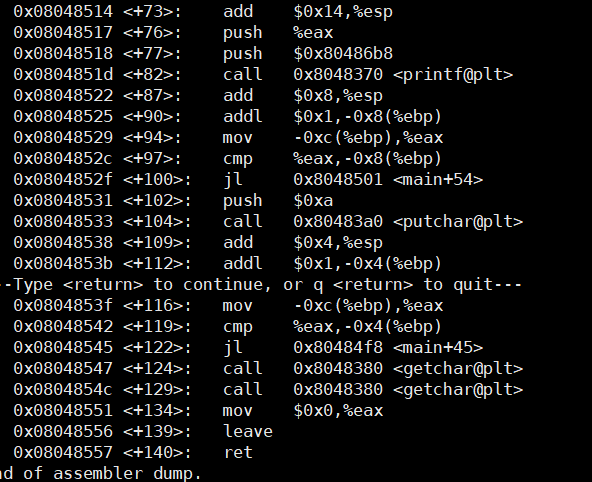
**소스코드**

|  |
| --- |
| **#include<stdio.h>**  **int snailMatrix(int b, int a, int c, int x, int y);**  **int main() {**  **int a,b,c;**  **printf("N : ");**  **scanf("%d",&c);**  **for(a=0; a<c; a++){**  **for(b=0; b<c; b++){**  **printf("%4d",snailMatrix(b,a,c,0,1));**  **}**  **putchar(10);**  **}**  **getchar();**  **getchar();**  **}**  **int snailMatrix(int b, int a, int c, int x, int y){**  **int i;**  **if(c==1){**  **i=y;**  **return i;**  **}**  **else{**  **if(x!=a){**  **if((x+c-1)!=b){**  **if(a!=x+c-1){**  **if(b!=x){**  **i= snailMatrix(b,a,c-2,x+1,4\*c+y-4);**  **return i;**  **}**  **}**    **i=(c\*2+x);**  **i=(2\*i+y);**  **return i-b-a-4;**    **}**  **}**  **i=b+y;**  **return (i-2\*x)+a;**  **}**    **}** |

**출력결과**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

****

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**<조사과제>**

비트: 컴퓨터가 데이터를 처리하기 위해 사용하는 데이터의 최소단위

-> 2진수의 값(0,1)을 단 하나만 저장할 수 있음

바이트: 8개비트가 모여서 구성, 한문자를 표현할 수 있는 최소단위

데이터를 메모리에 저장할 때 바이트 단위로 나눠서 저장하는데 컴퓨터가 저장하는 데이터는 32비트(4바이트), 64비트(8바이트)로 구성됨-> 연속되는 바이트를 순서대로 저장해야하는데 방법이 두가지가 있음

|  |  |
| --- | --- |
| 빅엔디안(big endian) | 리틀엔디안(little endian) |
| 낮은 주소에 데이터의 높은 바이트부터 저장하는 방식  -평소 우리가 사용하는 선형방식  -메모리에 저장된 순서 그대로 읽을 수 있고 이해하기 쉬움  -데이터의 각 바이트를 배열처럼 취급할 때는 빅엔디안이 적합함  Ex) 0x12345678  -> 0x12,0x34,0x56,0x78 | 낮은 주소에 데이터의 낮은 바이트부터 저장하는 방식  -평소 우리가 사용하는 선형방식과 반대인 거꾸로 읽어야함  -물리적으로 데이터를 조작하거나 산술 연산을 수행할 때 더 효율적임  -인텔기반의 윈도우는 리틀엔디안방식을 사용하지만 네트워크를 통해 데이터를 전송할때는 빅엔디안 방식이 사용됨  Ex) 0x12345678  -> 0x78,0x56,0x34,0x12 |