Mystery of return in main function

Who runs this main ?

Program 

```
int main(int argc, char  *argv[] )
{

Char buffer[64];

strcpy(buffer,argv[1]);

return 0;
}
```

int __libc_start_main()

Program

```
int __libc_start_main()
{
.....
main();
.....
}
```

call

Return to
libc_start_main()

```
int main(int argc,char *argv[])
{

char buffer[64];

strcpy(buffer,argv[1]);

return 0;
}
```

int __libc_start_main()

Stack

Program

```
int __libc_start_main()
{
.....
main();
0x7f3451fe
.....
}
```

calls

```
int main(int argc,char *argv[])
{

char buffer[64];

strcpy(buffer,argv[1]);

return 0;
}
```

Return to
libc_start_main()

Return to libc_start_main()

[address]: 0x7f3451fe

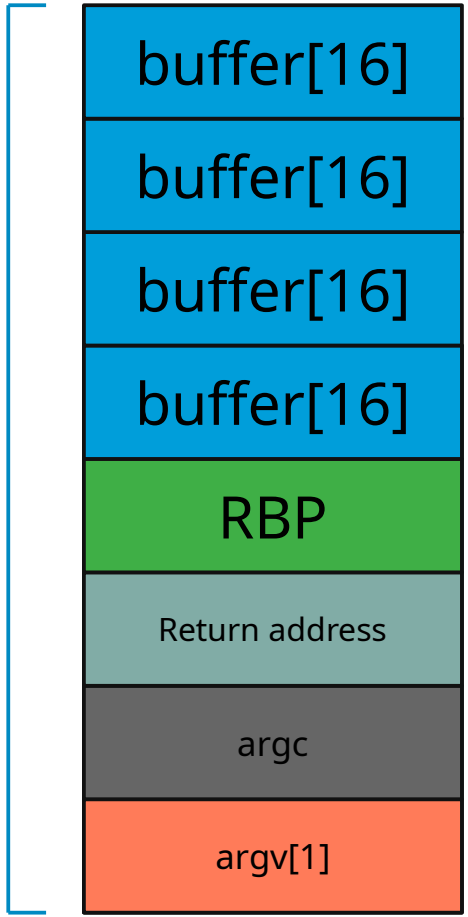
1

2

3

program

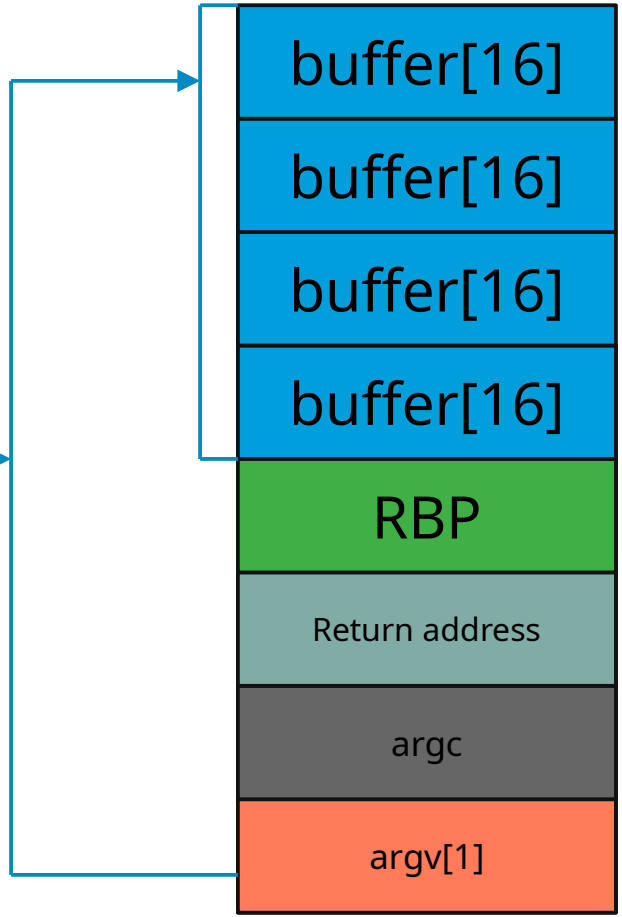
```
Int main(int argc,char *argv[])
{

Char buffer[64];

strcpy(buffer,argv[1]);

Return 0;
}
```



Stack memory

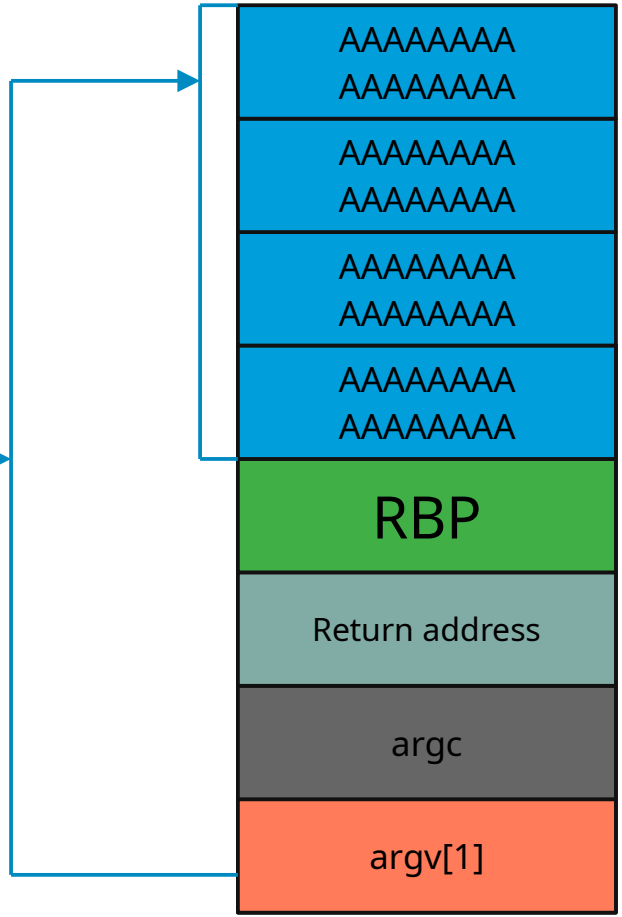
program

```
int main(int argc,char *argv[]){
char buffer[64];

strcpy ( buffer, argv[1] );

return 0;
}
```



program

```
Int main(int argc,char *argv[]){
Char buffer[64];
strcpy(buffer,argv[1]);
Return 0;
}
```

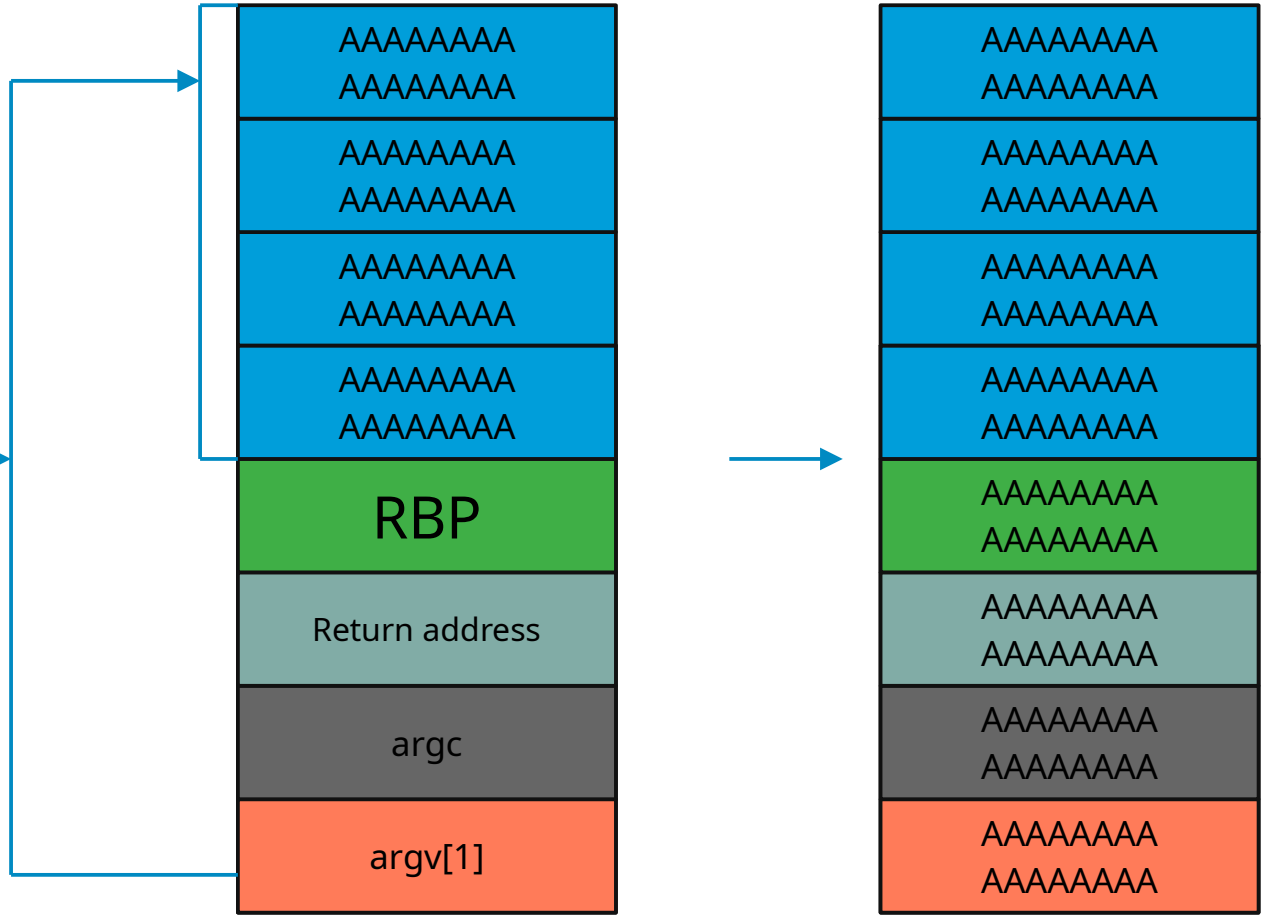


Argv = A 64 times
(64 bytes)

program

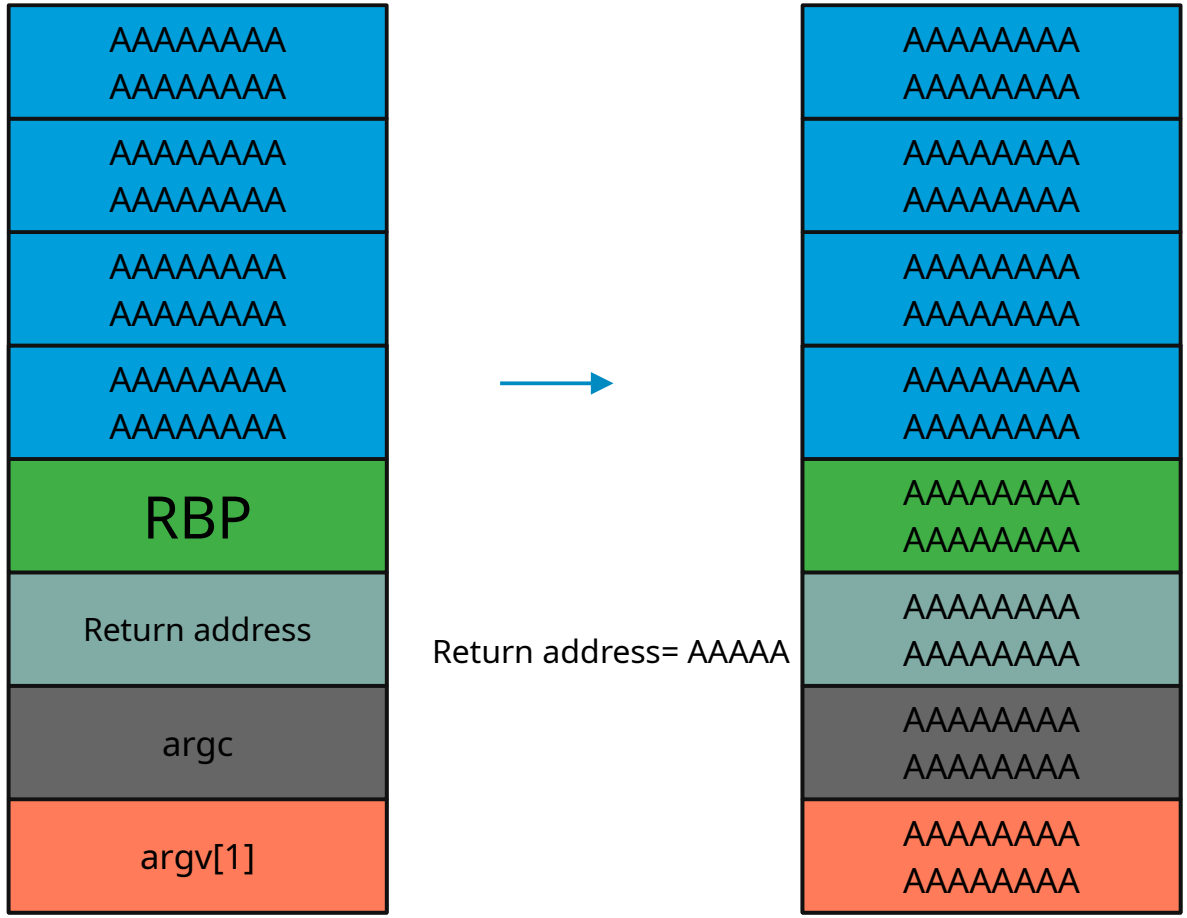
```
Int main(int argc,char *argv[]){
Char buffer[64];
strcpy(buffer,argv[1]);
Return 0;
}
```

Argv = A 100 times
(64 bytes)



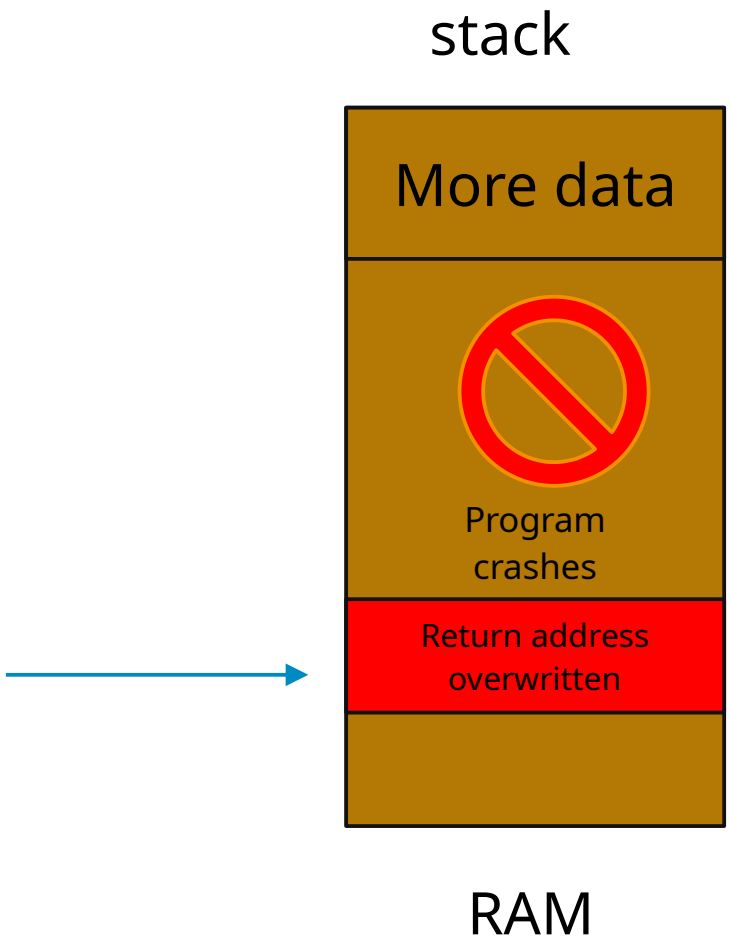
RIP=AAAAA

Where is this AAAA address ?
Where should I go?



This situation is called buffer overflow

When we have a control to the EIP regiter to redirect the program to anywhere else in memory this is called stack based buffer overflow exploit



Hacker

hacked

3

stack

1

2

Malicious code

Malicous code

New malicous code

RAM

Binary inside the computer