Code Profiling and Optimization

Day 2

Memory layout and using Debugging tools

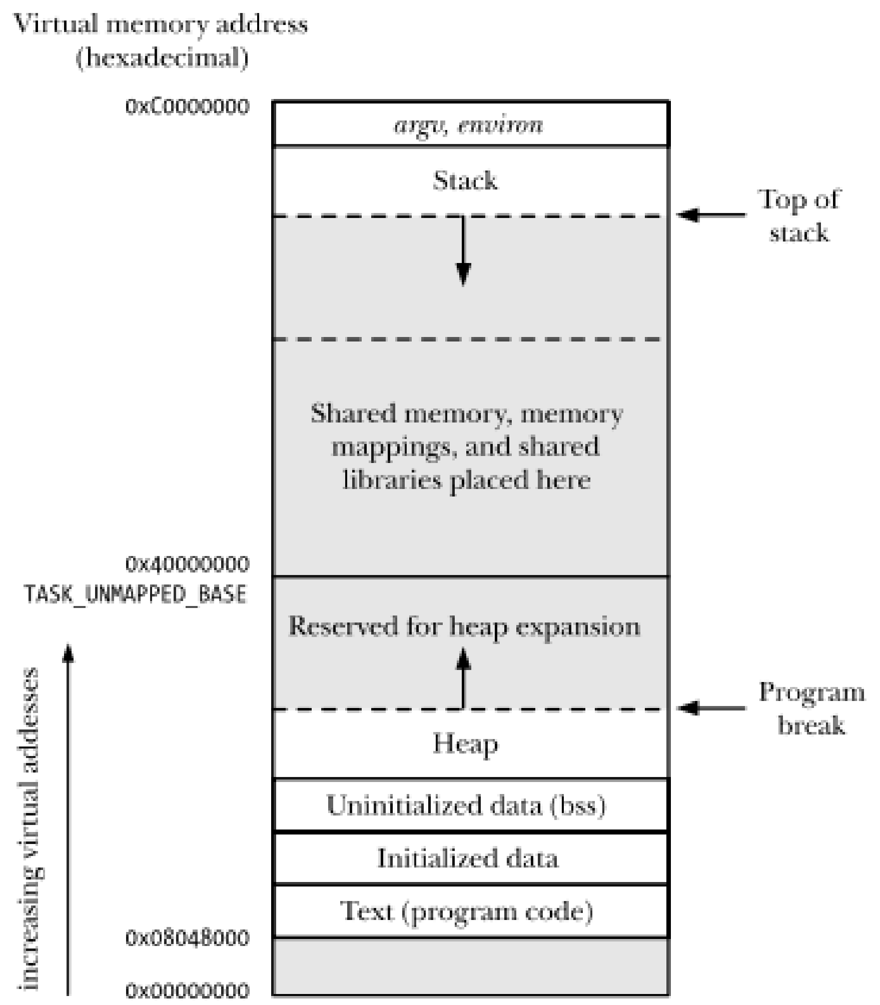


Figure 48-2: Locations of shared memory, memory mappings, and shared libraries (x86-32)

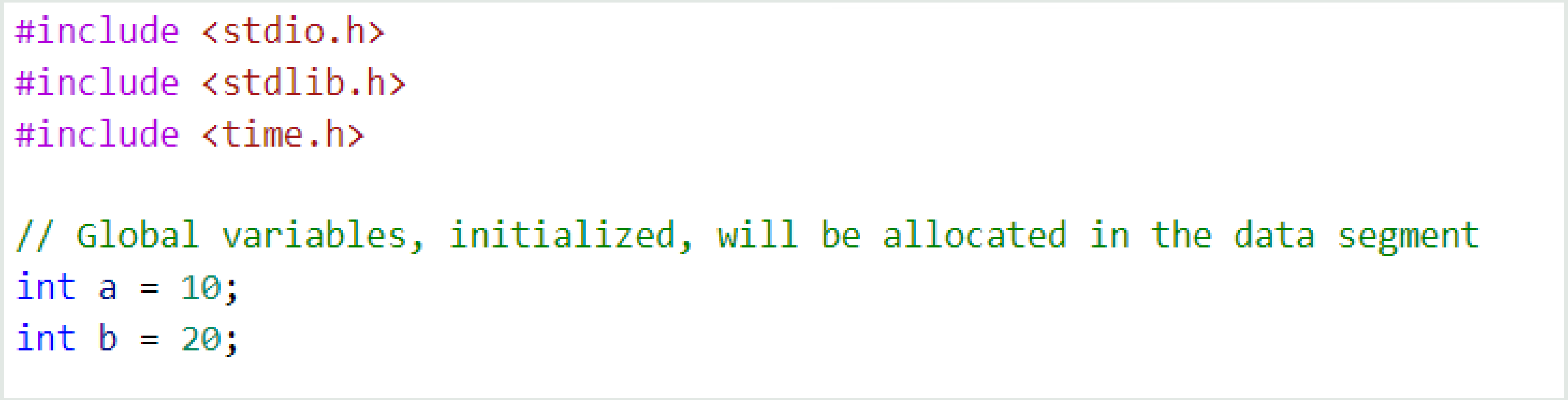
Memory Layout

Sections

- `.text`
- `.data`
- `.bss`
- `.rodata`
- Stack
- Heap

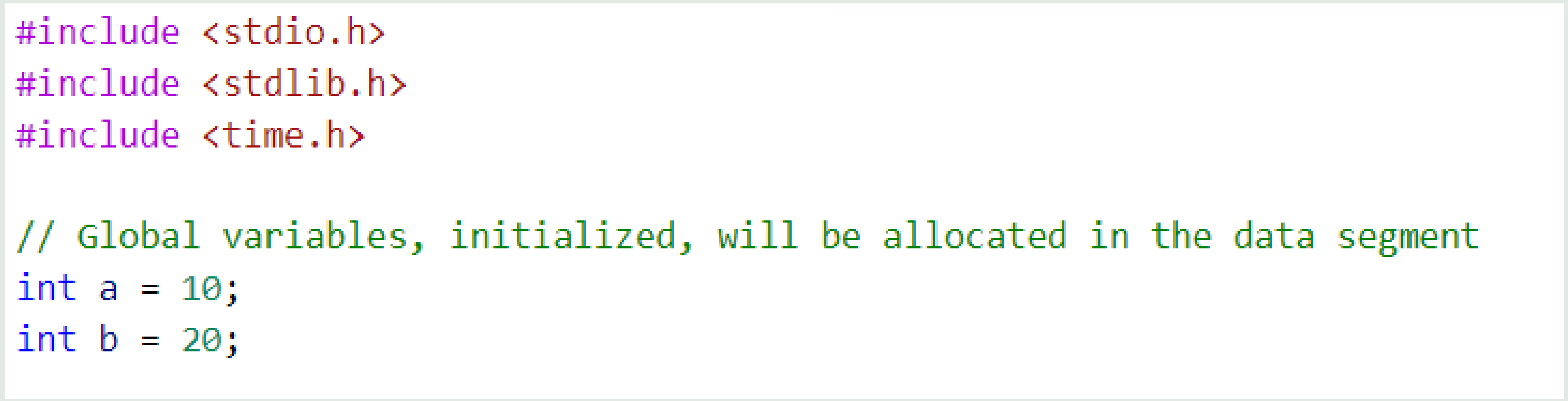
.data

- Global initialized variables
- For e.g.



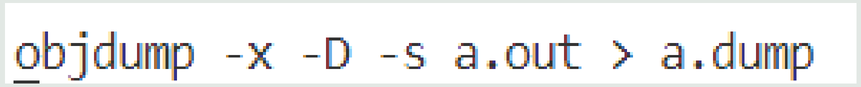
```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

// Global variables, initialized, will be allocated in the data segment
int a = 10;
int b = 20;
```

.data

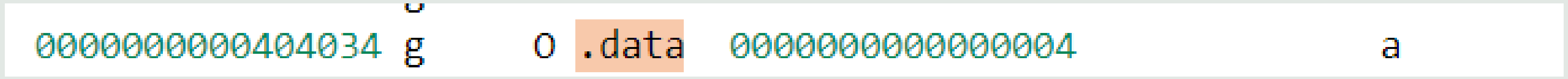


```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

// Global variables, initialized, will be allocated in the data segment
int a = 10;
int b = 20;
```

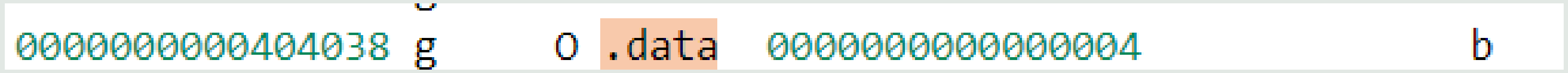
- Global initialized variables
- For e.g.



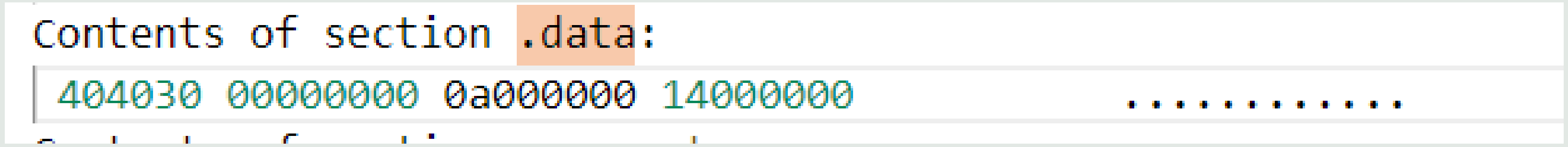
```
objdump -x -D -s a.out > a.dump
```



```
0000000000404034 g     O .data  0000000000000004              a
```



```
0000000000404038 g     O .data  0000000000000004              b
```



```
Contents of section .data:
 404030 00000000 0a000000 14000000           ............
```

.bss

- Global uninitialized variables
- For e.g.

```c
// Global variables, uninitialized, will be allocated in the bss segment
int c;
int d;
```

.bss

```
// Global variables, uninitialized, will be allocated in the bss segment
int c;
int d;
```

- Global uninitialized variables
- For e.g.

```
objdump -x -D -s a.out > a.dump
```

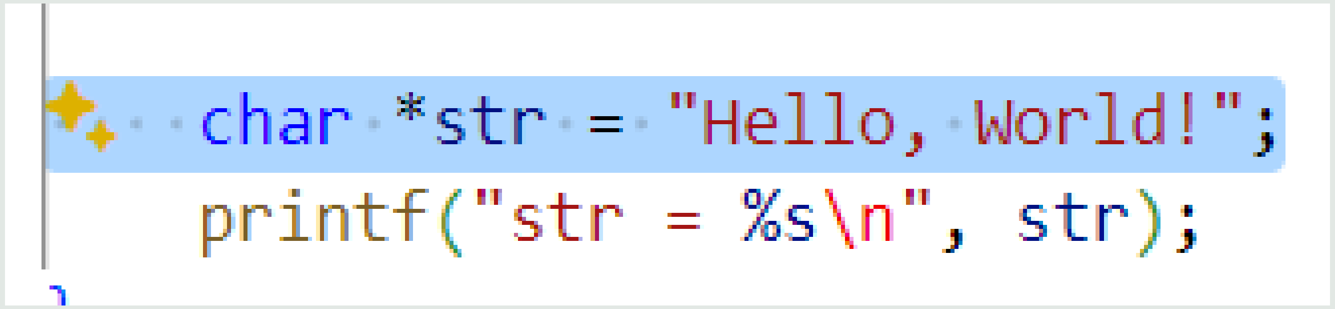
```
24 .bss          0000000c  000000000040403c  000000000040403c  0000303c  2**2
```

```
0000000000404040 g     O .bss 0000000000000004              c
```

```
0000000000404044 g     O .bss 0000000000000004              d
```

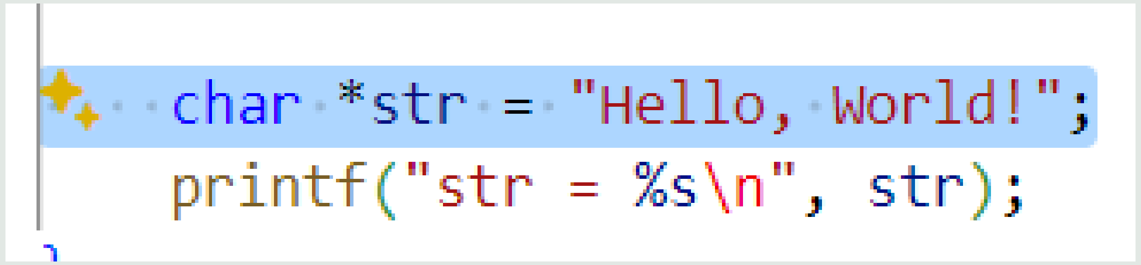
.rodata

- Read only data; Note if we try to modify .rodata, it will generate seg fault!
- For e.g.



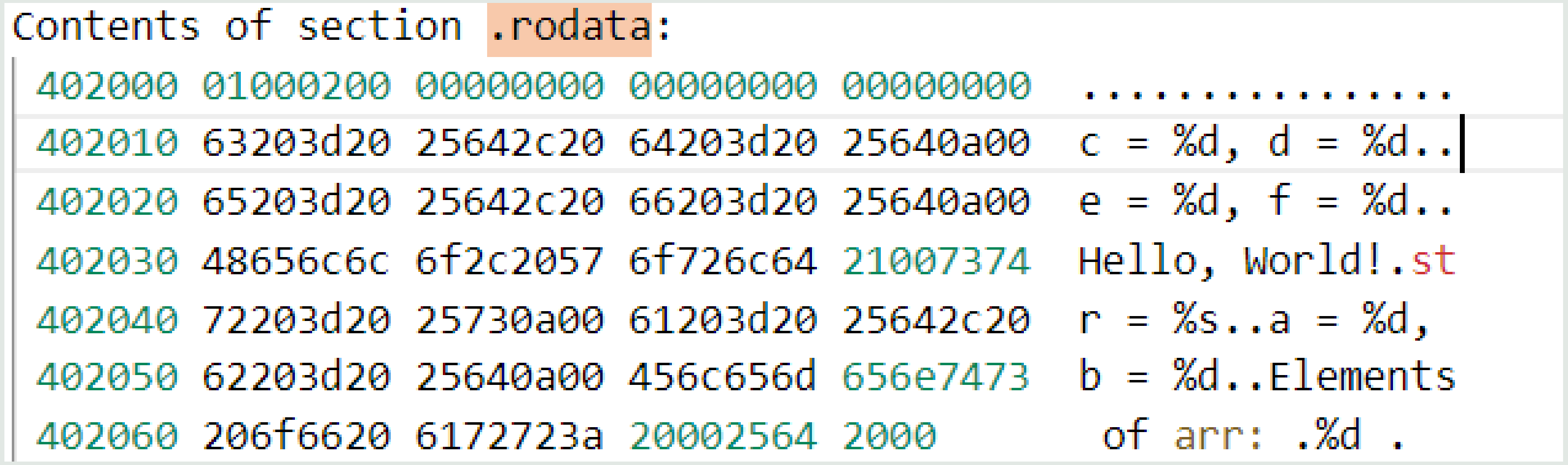
```
    char *str = "Hello, World!";
    printf("str = %s\n", str);
}
```

.rodata



```
char *str = "Hello, World!";
printf("str = %s\n", str);
```

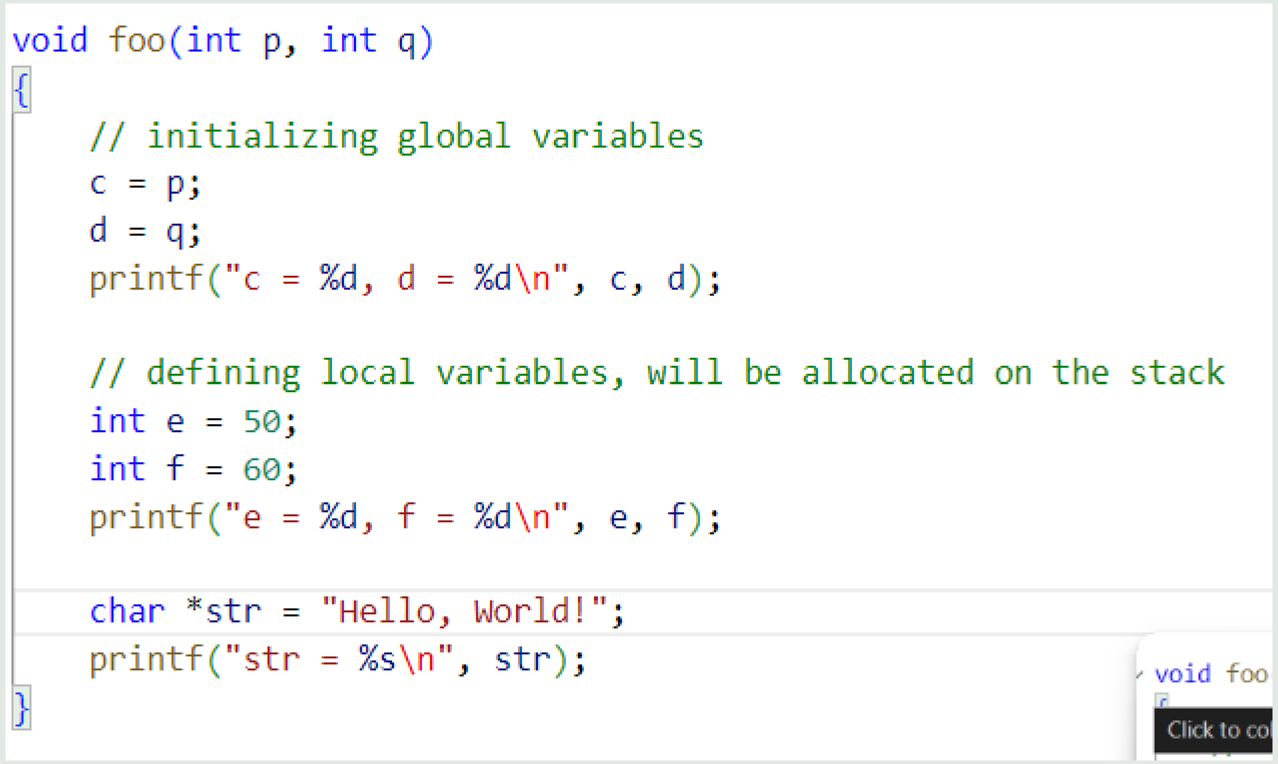
- Read only data; Note if we try to modify .rodata, it will generate seg fault!
- For e.g.



```
Contents of section .rodata:
 402000 01000200 00000000 00000000 00000000  ................
 402010 63203d20 25642c20 64203d20 25640a00  c = %d, d = %d..
 402020 65203d20 25642c20 66203d20 25640a00  e = %d, f = %d..
 402030 48656c6c 6f2c2057 6f726c64 21007374  Hello, World!.st
 402040 72203d20 25730a00 61203d20 25642c20  r = %s..a = %d,
 402050 62203d20 25640a00 456c656d 656e7473  b = %d..Elements
 402060 206f6620 6172723a 20002564 2000       of arr: .%d .
```

.text

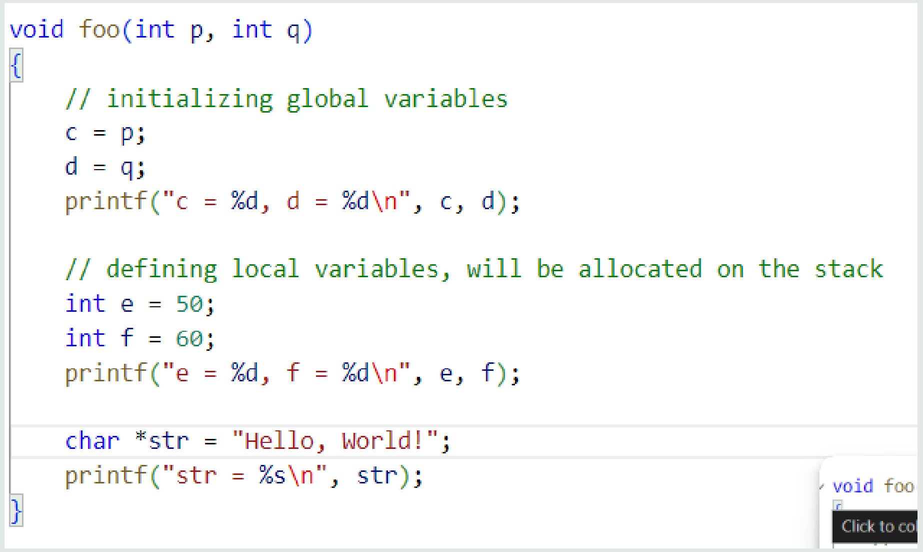
- Code segment, instructions to be executed
- For e.g.



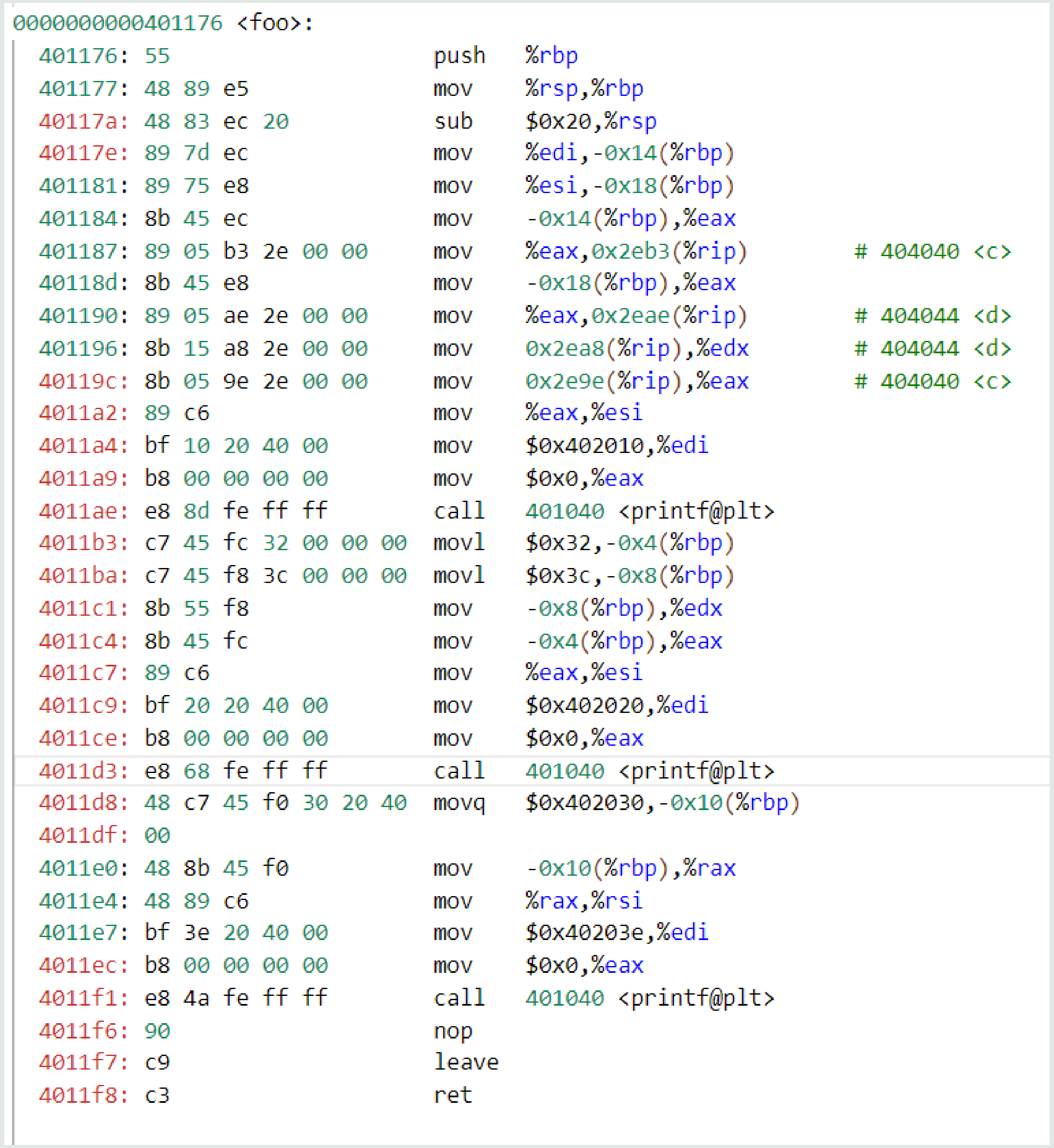
```
void foo(int p, int q)
{
    // initializing global variables
    c = p;
    d = q;
    printf("c = %d, d = %d\n", c, d);

    // defining local variables, will be allocated on the stack
    int e = 50;
    int f = 60;
    printf("e = %d, f = %d\n", e, f);

    char *str = "Hello, World!";
    printf("str = %s\n", str);
}
```

.text

- Code segment, instructions to be executed
- For e.g.



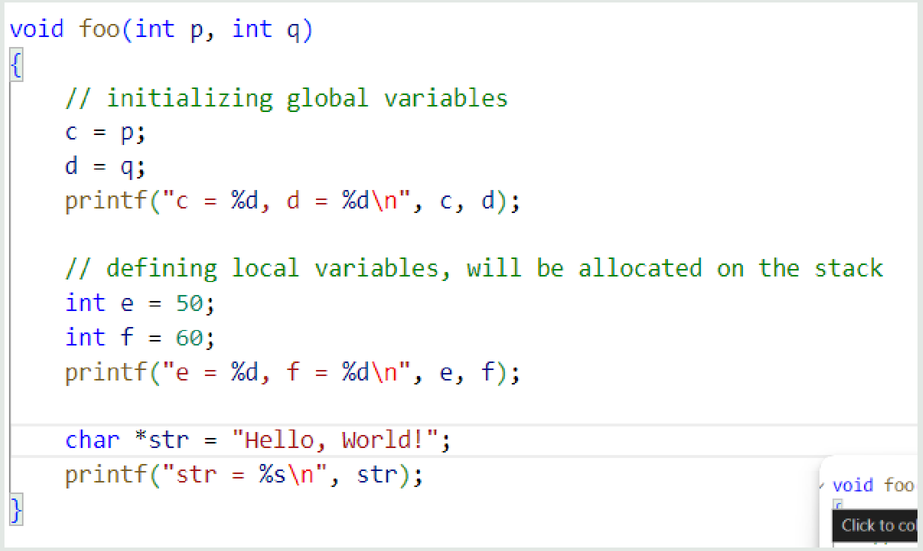
```c
void foo(int p, int q)
{
    // initializing global variables
    c = p;
    d = q;
    printf("c = %d, d = %d\n", c, d);

    // defining local variables, will be allocated on the stack
    int e = 50;
    int f = 60;
    printf("e = %d, f = %d\n", e, f);

    char *str = "Hello, World!";
    printf("str = %s\n", str);
}
```



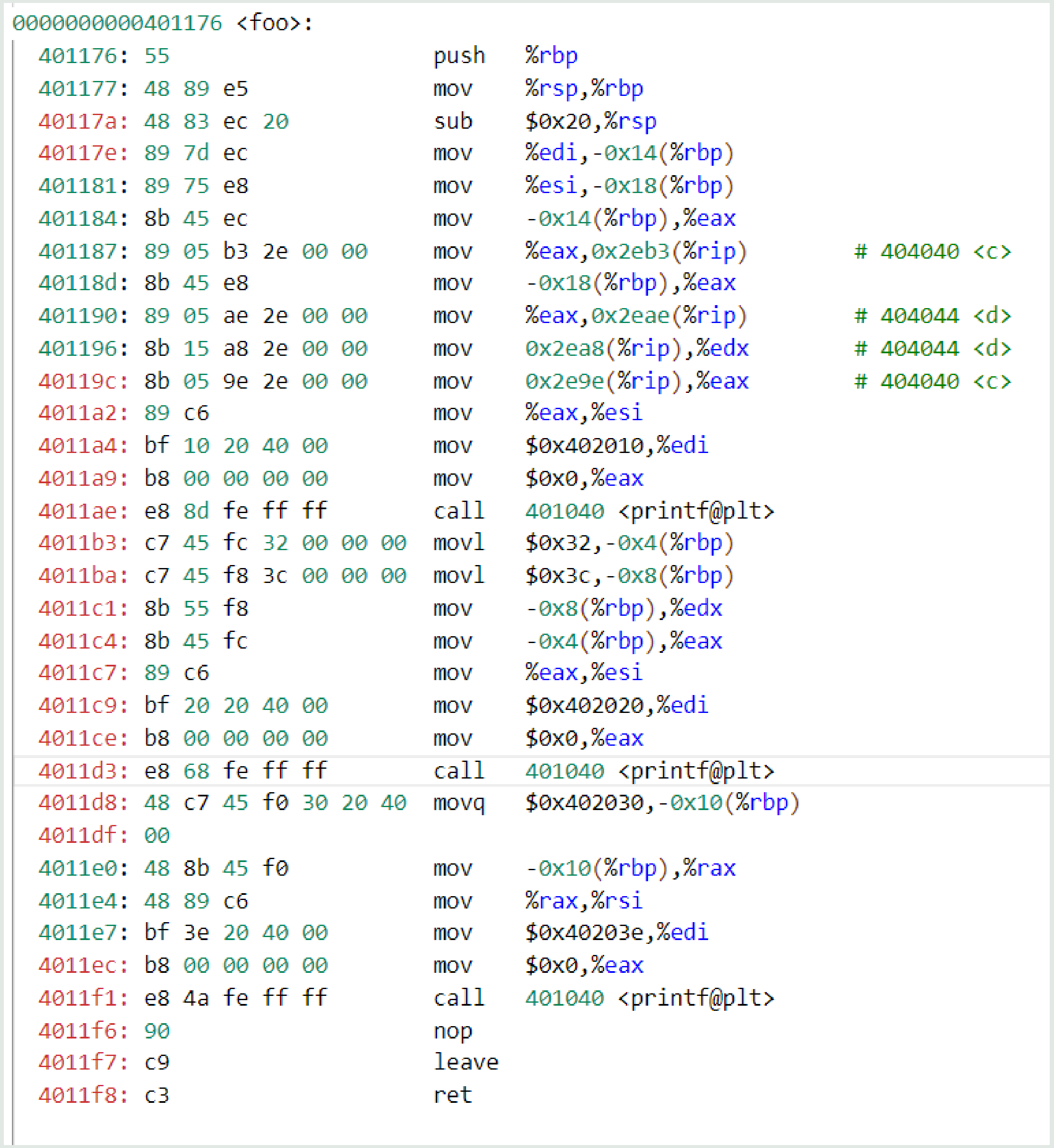
```
0000000000401176 <foo>:
  401176: 55                    push   %rbp
  401177: 48 89 e5              mov    %rsp,%rbp
  40117a: 48 83 ec 20           sub    $0x20,%rsp
  40117e: 89 7d ec              mov    %edi,-0x14(%rbp)
  401181: 89 75 e8              mov    %esi,-0x18(%rbp)
  401184: 8b 45 ec              mov    -0x14(%rbp),%eax
  401187: 89 05 b3 2e 00 00     mov    %eax,0x2eb3(%rip)        # 404040 <c>
  40118d: 8b 45 e8              mov    -0x18(%rbp),%eax
  401190: 89 05 ae 2e 00 00     mov    %eax,0x2eae(%rip)        # 404044 <d>
  401196: 8b 15 a8 2e 00 00     mov    0x2ea8(%rip),%edx        # 404044 <d>
  40119c: 8b 05 9e 2e 00 00     mov    0x2e9e(%rip),%eax        # 404040 <c>
  4011a2: 89 c6                 mov    %eax,%esi
  4011a4: bf 10 20 40 00        mov    $0x402010,%edi
  4011a9: b8 00 00 00 00        mov    $0x0,%eax
  4011ae: e8 8d fe ff ff        call   401040 <printf@plt>
  4011b3: c7 45 fc 32 00 00 00  movl   $0x32,-0x4(%rbp)
  4011ba: c7 45 f8 3c 00 00 00  movl   $0x3c,-0x8(%rbp)
  4011c1: 8b 55 f8              mov    -0x8(%rbp),%edx
  4011c4: 8b 45 fc              mov    -0x4(%rbp),%eax
  4011c7: 89 c6                 mov    %eax,%esi
  4011c9: bf 20 20 40 00        mov    $0x402020,%edi
  4011ce: b8 00 00 00 00        mov    $0x0,%eax
  4011d3: e8 68 fe ff ff        call   401040 <printf@plt>
  4011d8: 48 c7 45 f0 30 20 40  movq   $0x402030,-0x10(%rbp)
  4011df: 00
  4011e0: 48 8b 45 f0           mov    -0x10(%rbp),%rax
  4011e4: 48 89 c6              mov    %rax,%rsi
  4011e7: bf 3e 20 40 00        mov    $0x40203e,%edi
  4011ec: b8 00 00 00 00        mov    $0x0,%eax
  4011f1: e8 4a fe ff ff        call   401040 <printf@plt>
  4011f6: 90                    nop
  4011f7: c9                    leave
  4011f8: c3                    ret
```

stack

- Local variables will be adjusted in stack
- For e.g.



```
void foo(int p, int q)
{
    // initializing global variables
    c = p;
    d = q;
    printf("c = %d, d = %d\n", c, d);

    // defining local variables, will be allocated on the stack
    int e = 50;
    int f = 60;
    printf("e = %d, f = %d\n", e, f);

    char *str = "Hello, World!";
    printf("str = %s\n", str);
}
```

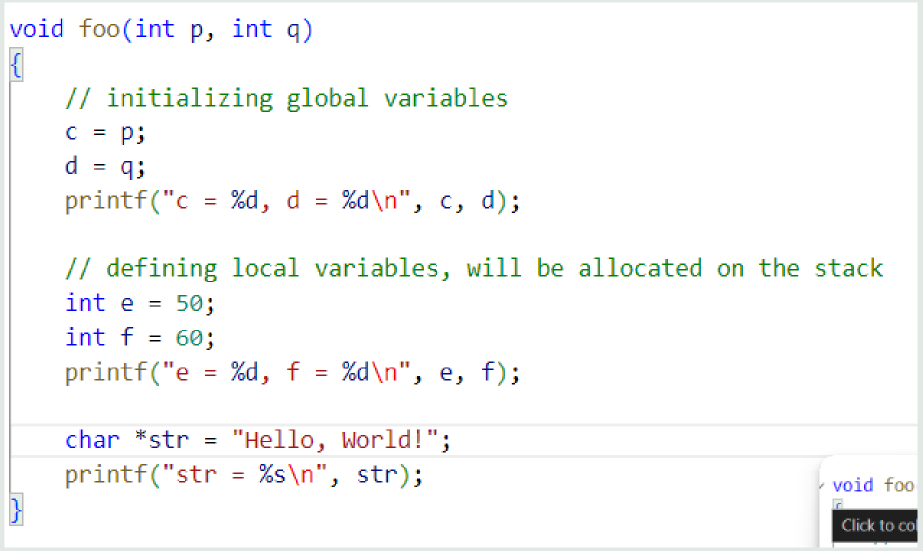


```
0000000000401176 <foo>:
  401176: 55                      push   %rbp
  401177: 48 89 e5                mov    %rsp,%rbp
  40117a: 48 83 ec 20             sub    $0x20,%rsp
  40117e: 89 7d ec                mov    %edi,-0x14(%rbp)
  401181: 89 75 e8                mov    %esi,-0x18(%rbp)
  401184: 8b 45 ec                mov    -0x14(%rbp),%eax
  401187: 89 05 b3 2e 00 00       mov    %eax,0x2eb3(%rip)        # 404040 <c>
  40118d: 8b 45 e8                mov    -0x18(%rbp),%eax
  401190: 89 05 ae 2e 00 00       mov    %eax,0x2eae(%rip)        # 404044 <d>
  401196: 8b 15 a8 2e 00 00       mov    0x2ea8(%rip),%edx        # 404044 <d>
  40119c: 8b 05 9e 2e 00 00       mov    0x2e9e(%rip),%eax        # 404040 <c>
  4011a2: 89 c6                   mov    %eax,%esi
  4011a4: bf 10 20 40 00          mov    $0x402010,%edi
  4011a9: b8 00 00 00 00          mov    $0x0,%eax
  4011ae: e8 8d fe ff ff          call   401040 <printf@plt>
  4011b3: c7 45 fc 32 00 00 00    movl   $0x32,-0x4(%rbp)
  4011ba: c7 45 f8 3c 00 00 00    movl   $0x3c,-0x8(%rbp)
  4011c1: 8b 55 f8                mov    -0x8(%rbp),%edx
  4011c4: 8b 45 fc                mov    -0x4(%rbp),%eax
  4011c7: 89 c6                   mov    %eax,%esi
  4011c9: bf 20 20 40 00          mov    $0x402020,%edi
  4011ce: b8 00 00 00 00          mov    $0x0,%eax
  4011d3: e8 68 fe ff ff          call   401040 <printf@plt>
  4011d8: 48 c7 45 f0 30 20 40    movq   $0x402030,-0x10(%rbp)
  4011df: 00
  4011e0: 48 8b 45 f0             mov    -0x10(%rbp),%rax
  4011e4: 48 89 c6                mov    %rax,%rsi
  4011e7: bf 3e 20 40 00          mov    $0x40203e,%edi
  4011ec: b8 00 00 00 00          mov    $0x0,%eax
  4011f1: e8 4a fe ff ff          call   401040 <printf@plt>
  4011f6: 90                      nop
  4011f7: c9                      leave
  4011f8: c3                      ret
```

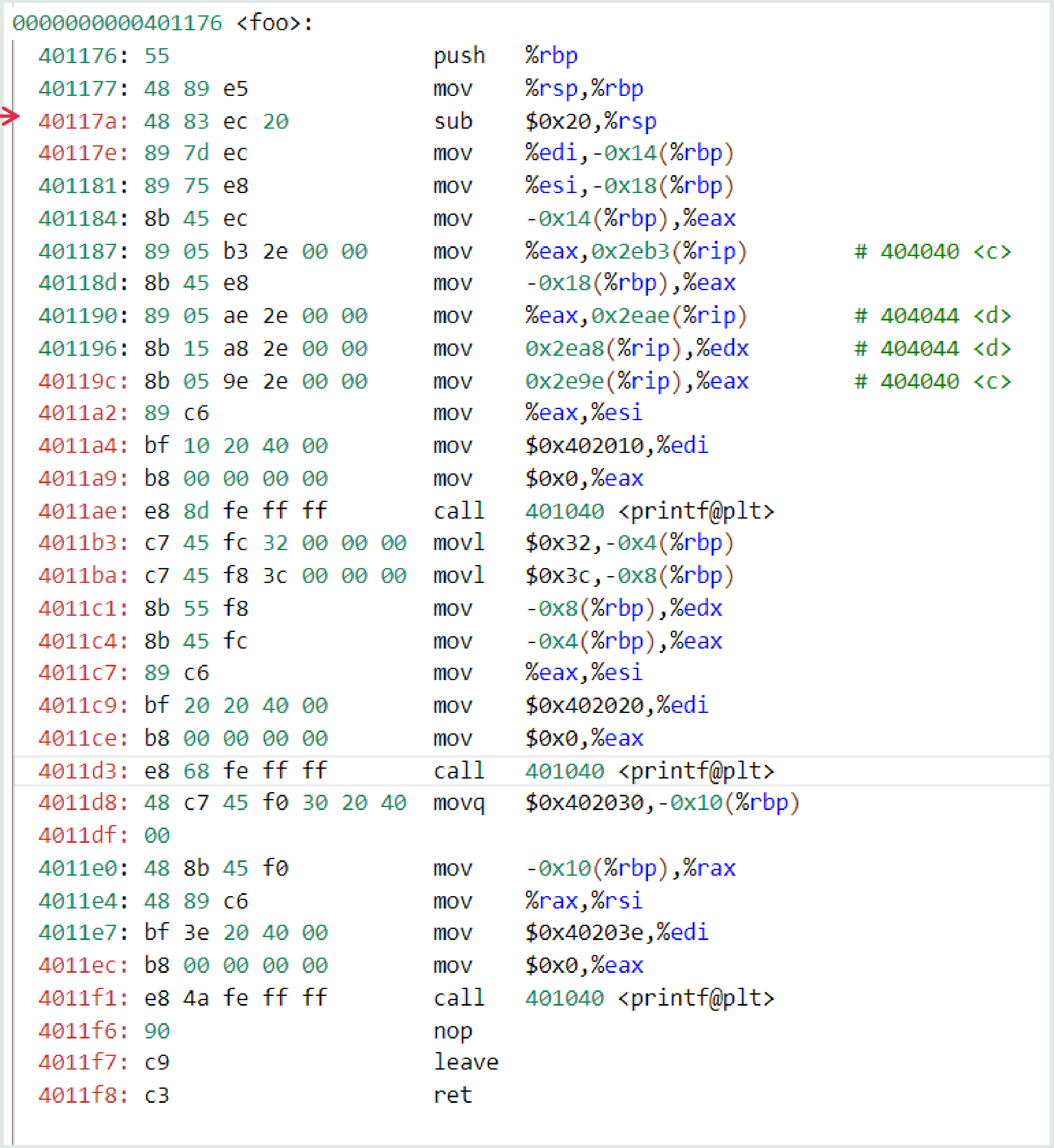
Allocates 32 bytes on the stack for local variables.

stack

- Local variables will be adjusted in stack
- For e.g.



```
void foo(int p, int q)
{
    // initializing global variables
    c = p;
    d = q;
    printf("c = %d, d = %d\n", c, d);

    // defining local variables, will be allocated on the stack
    int e = 50;
    int f = 60;
    printf("e = %d, f = %d\n", e, f);

    char *str = "Hello, World!";
    printf("str = %s\n", str);
}
```



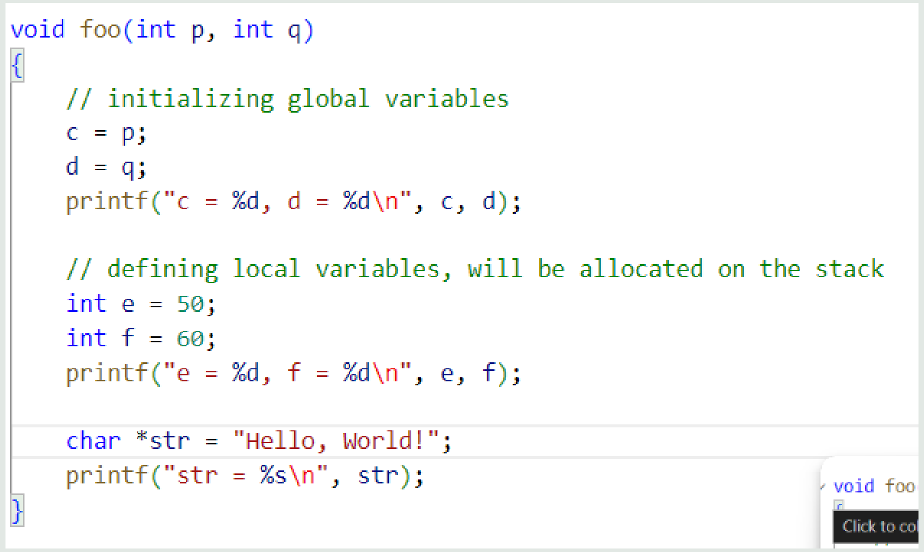
```
0000000000401176 <foo>:
  401176: 55                      push   %rbp
  401177: 48 89 e5                mov    %rsp,%rbp
  40117a: 48 83 ec 20             sub    $0x20,%rsp
  40117e: 89 7d ec                mov    %edi,-0x14(%rbp)
  401181: 89 75 e8                mov    %esi,-0x18(%rbp)
  401184: 8b 45 ec                mov    -0x14(%rbp),%eax
  401187: 89 05 b3 2e 00 00       mov    %eax,0x2eb3(%rip)        # 404040 <c>
  40118d: 8b 45 e8                mov    -0x18(%rbp),%eax
  401190: 89 05 ae 2e 00 00       mov    %eax,0x2eae(%rip)        # 404044 <d>
  401196: 8b 15 a8 2e 00 00       mov    0x2ea8(%rip),%edx        # 404044 <d>
  40119c: 8b 05 9e 2e 00 00       mov    0x2e9e(%rip),%eax        # 404040 <c>
  4011a2: 89 c6                   mov    %eax,%esi
  4011a4: bf 10 20 40 00          mov    $0x402010,%edi
  4011a9: b8 00 00 00 00          mov    $0x0,%eax
  4011ae: e8 8d fe ff ff          call   401040 <printf@plt>
  4011b3: c7 45 fc 32 00 00 00    movl   $0x32,-0x4(%rbp)
  4011ba: c7 45 f8 3c 00 00 00    movl   $0x3c,-0x8(%rbp)
  4011c1: 8b 55 f8                mov    -0x8(%rbp),%edx
  4011c4: 8b 45 fc                mov    -0x4(%rbp),%eax
  4011c7: 89 c6                   mov    %eax,%esi
  4011c9: bf 20 20 40 00          mov    $0x402020,%edi
  4011ce: b8 00 00 00 00          mov    $0x0,%eax
  4011d3: e8 68 fe ff ff          call   401040 <printf@plt>
  4011d8: 48 c7 45 f0 30 20 40    movq   $0x402030,-0x10(%rbp)
  4011df: 00
  4011e0: 48 8b 45 f0             mov    -0x10(%rbp),%rax
  4011e4: 48 89 c6                mov    %rax,%rsi
  4011e7: bf 3e 20 40 00          mov    $0x40203e,%edi
  4011ec: b8 00 00 00 00          mov    $0x0,%eax
  4011f1: e8 4a fe ff ff          call   401040 <printf@plt>
  4011f6: 90                      nop
  4011f7: c9                      leave
  4011f8: c3                      ret
```

stack

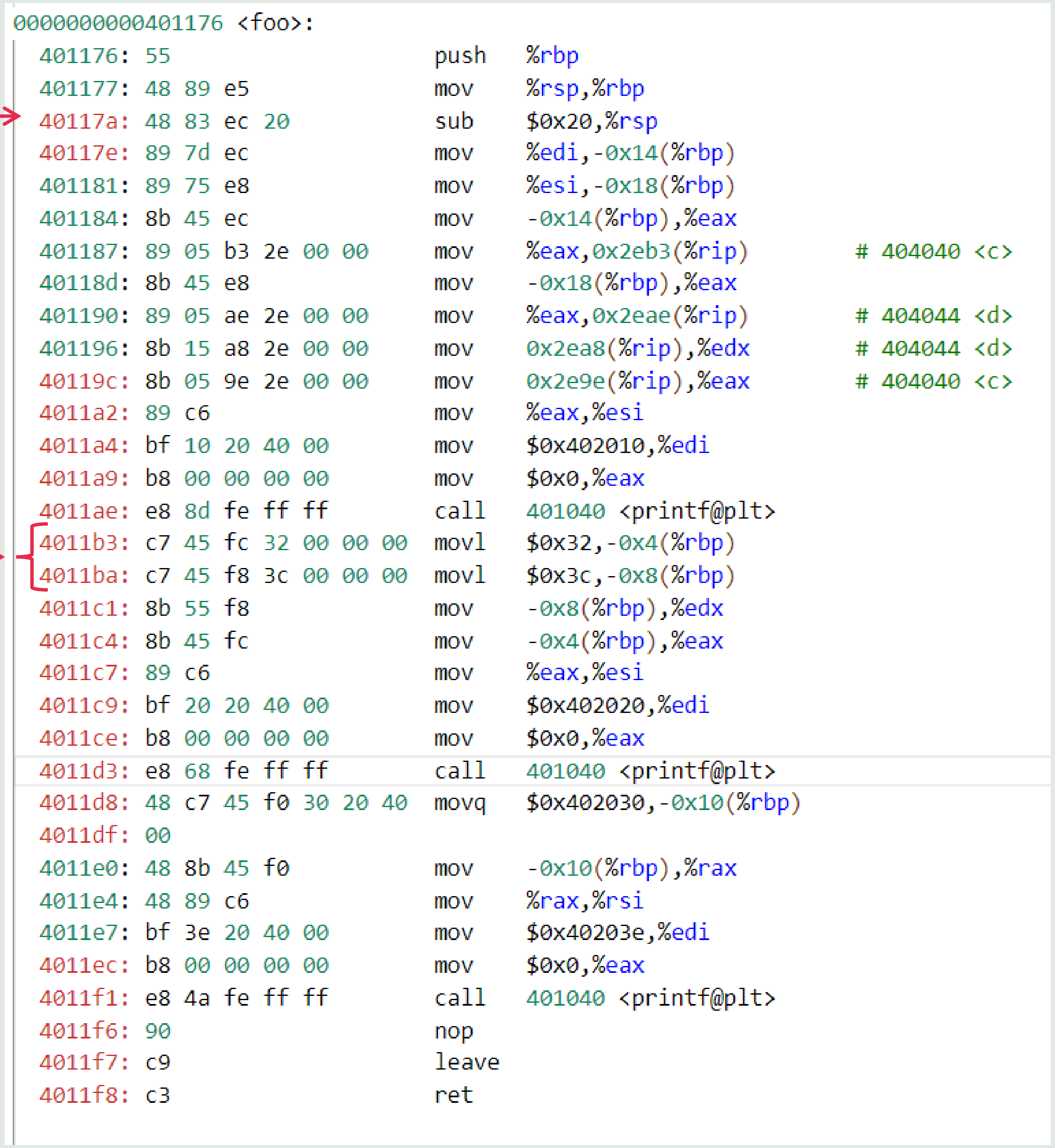
- Local variables will be adjusted in stack
- For e.g.

Allocates 32 bytes on the stack for local variables.

Stores the values of p and q.



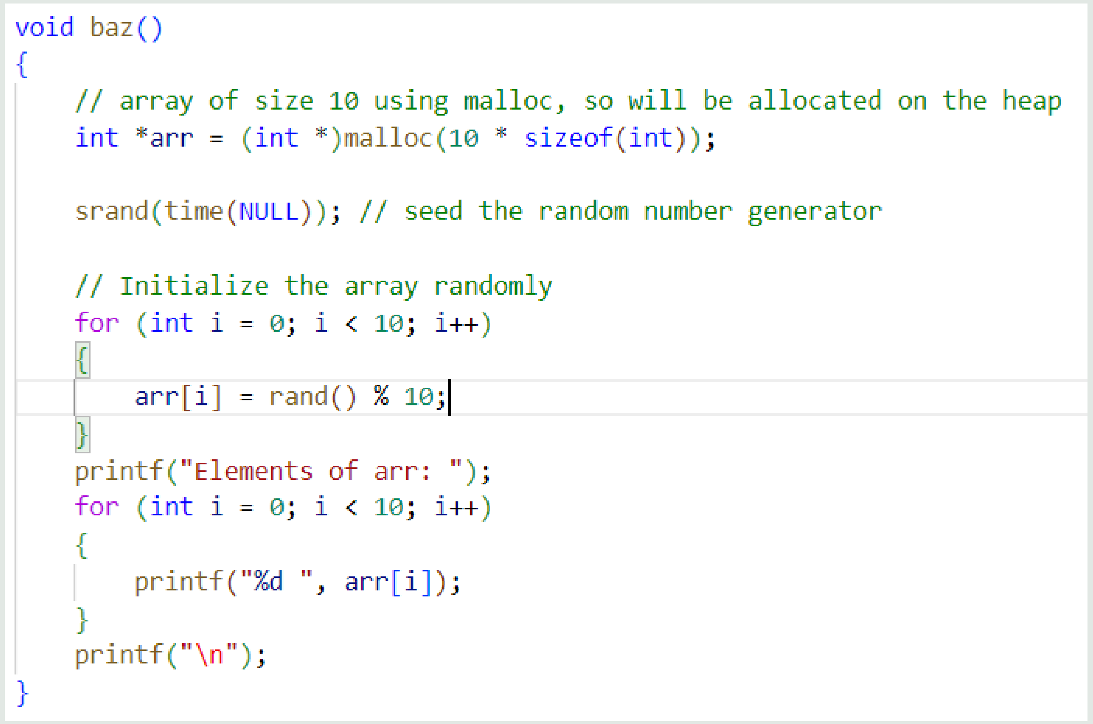
```c
void foo(int p, int q)
{
    // initializing global variables
    c = p;
    d = q;
    printf("c = %d, d = %d\n", c, d);

    // defining local variables, will be allocated on the stack
    int e = 50;
    int f = 60;
    printf("e = %d, f = %d\n", e, f);

    char *str = "Hello, World!";
    printf("str = %s\n", str);
}
```



```
0000000000401176 <foo>:
  401176: 55                      push   %rbp
  401177: 48 89 e5                mov    %rsp,%rbp
  40117a: 48 83 ec 20             sub    $0x20,%rsp
  40117e: 89 7d ec                mov    %edi,-0x14(%rbp)
  401181: 89 75 e8                mov    %esi,-0x18(%rbp)
  401184: 8b 45 ec                mov    -0x14(%rbp),%eax
  401187: 89 05 b3 2e 00 00       mov    %eax,0x2eb3(%rip)        # 404040 <c>
  40118d: 8b 45 e8                mov    -0x18(%rbp),%eax
  401190: 89 05 ae 2e 00 00       mov    %eax,0x2eae(%rip)        # 404044 <d>
  401196: 8b 15 a8 2e 00 00       mov    0x2ea8(%rip),%edx        # 404044 <d>
  40119c: 8b 05 9e 2e 00 00       mov    0x2e9e(%rip),%eax        # 404040 <c>
  4011a2: 89 c6                   mov    %eax,%esi
  4011a4: bf 10 20 40 00          mov    $0x402010,%edi
  4011a9: b8 00 00 00 00          mov    $0x0,%eax
  4011ae: e8 8d fe ff ff          call   401040 <printf@plt>
  4011b3: c7 45 fc 32 00 00 00    movl   $0x32,-0x4(%rbp)
  4011ba: c7 45 f8 3c 00 00 00    movl   $0x3c,-0x8(%rbp)
  4011c1: 8b 55 f8                mov    -0x8(%rbp),%edx
  4011c4: 8b 45 fc                mov    -0x4(%rbp),%eax
  4011c7: 89 c6                   mov    %eax,%esi
  4011c9: bf 20 20 40 00          mov    $0x402020,%edi
  4011ce: b8 00 00 00 00          mov    $0x0,%eax
  4011d3: e8 68 fe ff ff          call   401040 <printf@plt>
  4011d8: 48 c7 45 f0 30 20 40    movq   $0x402030,-0x10(%rbp)
  4011df: 00
  4011e0: 48 8b 45 f0             mov    -0x10(%rbp),%rax
  4011e4: 48 89 c6                mov    %rax,%rsi
  4011e7: bf 3e 20 40 00          mov    $0x40203e,%edi
  4011ec: b8 00 00 00 00          mov    $0x0,%eax
  4011f1: e8 4a fe ff ff          call   401040 <printf@plt>
  4011f6: 90                      nop
  4011f7: c9                      leave
  4011f8: c3                      ret
```

heap

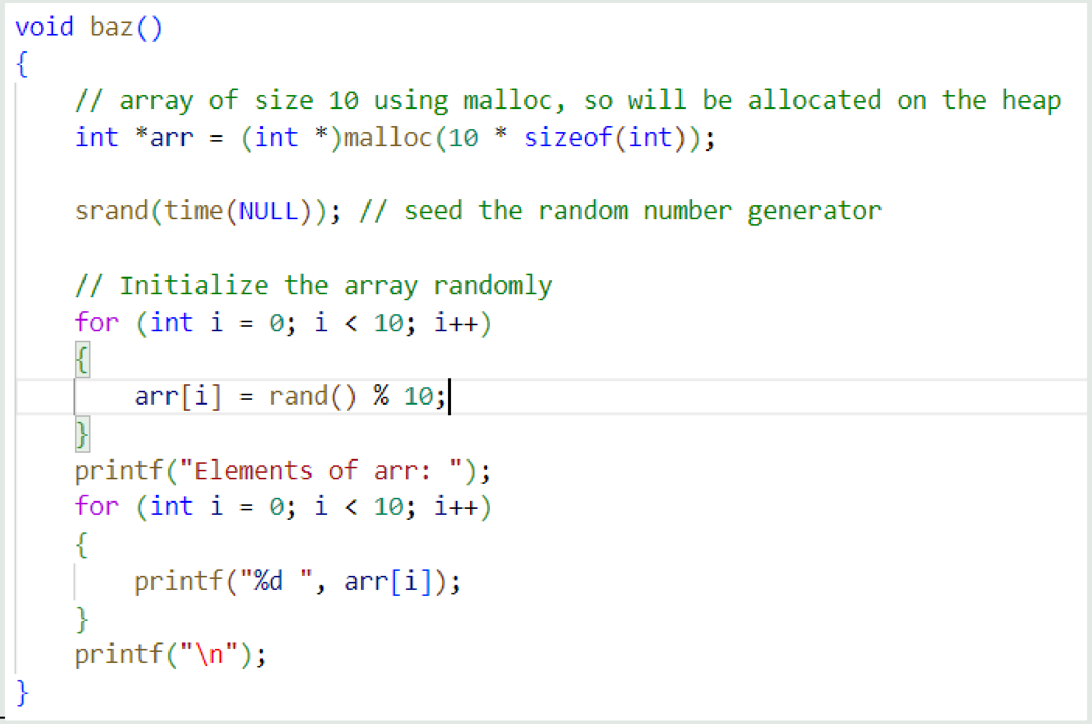
- Dynamically allocated variables will be in the heap
- For e.g.



```
void baz()
{
    // array of size 10 using malloc, so will be allocated on the heap
    int *arr = (int *)malloc(10 * sizeof(int));

    srand(time(NULL)); // seed the random number generator

    // Initialize the array randomly
    for (int i = 0; i < 10; i++)
    {
        arr[i] = rand() % 10;
    }
    printf("Elements of arr: ");
    for (int i = 0; i < 10; i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
```

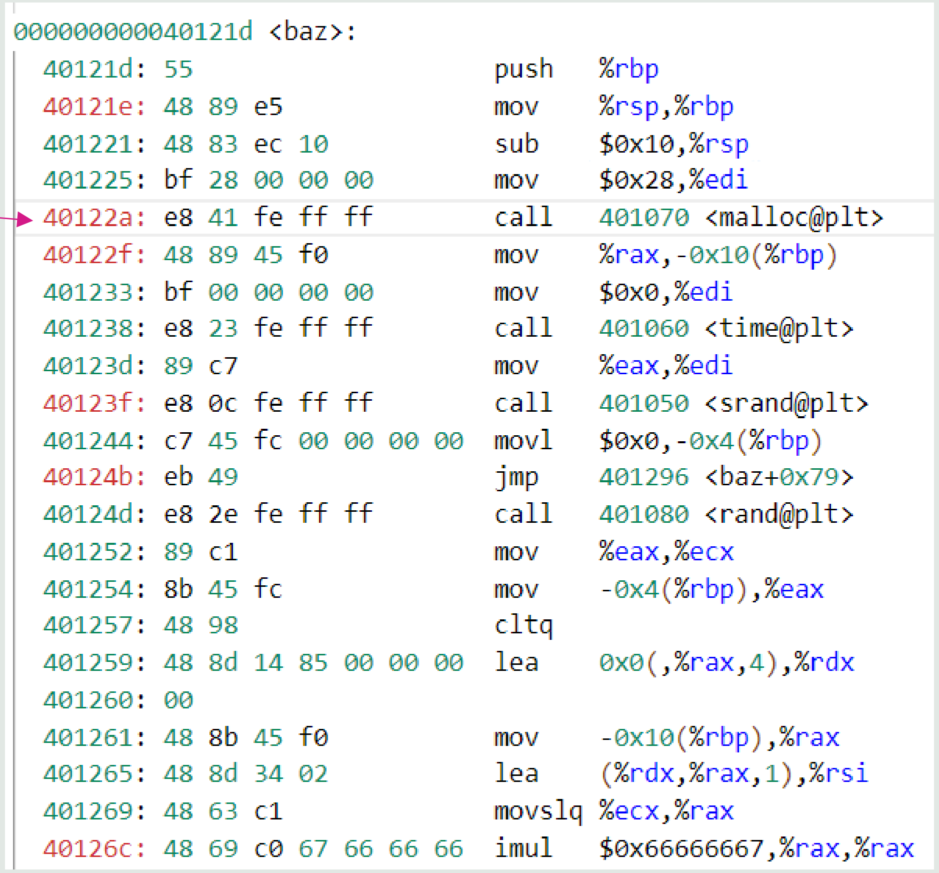
heap

- Dynamically allocated variables will be in the heap
- For e.g.



```c
void baz()
{
    // array of size 10 using malloc, so will be allocated on the heap
    int *arr = (int *)malloc(10 * sizeof(int));

    srand(time(NULL)); // seed the random number generator

    // Initialize the array randomly
    for (int i = 0; i < 10; i++)
    {
        arr[i] = rand() % 10;
    }
    printf("Elements of arr: ");
    for (int i = 0; i < 10; i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
```

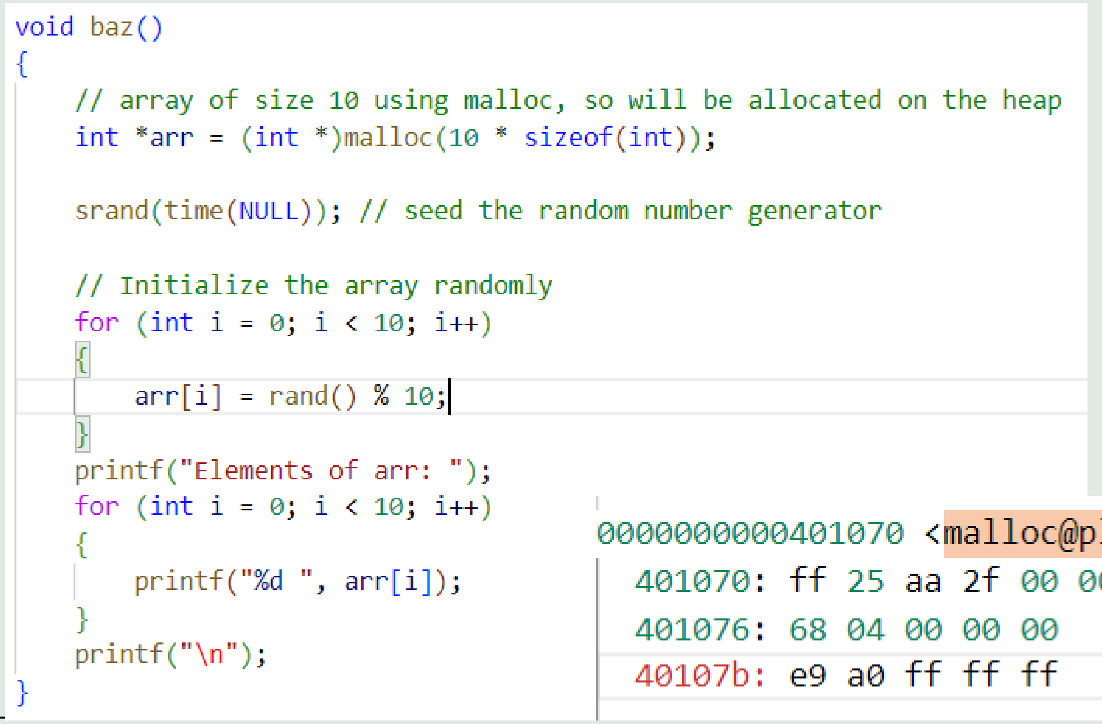
Call to malloc@plt



```
000000000040121d <baz>:
  40121d: 55                      push   %rbp
  40121e: 48 89 e5                mov    %rsp,%rbp
  401221: 48 83 ec 10             sub    $0x10,%rsp
  401225: bf 28 00 00 00          mov    $0x28,%edi
  40122a: e8 41 fe ff ff          call   401070 <malloc@plt>
  40122f: 48 89 45 f0             mov    %rax,-0x10(%rbp)
  401233: bf 00 00 00 00          mov    $0x0,%edi
  401238: e8 23 fe ff ff          call   401060 <time@plt>
  40123d: 89 c7                   mov    %eax,%edi
  40123f: e8 0c fe ff ff          call   401050 <srand@plt>
  401244: c7 45 fc 00 00 00 00    movl   $0x0,-0x4(%rbp)
  40124b: eb 49                   jmp    401296 <baz+0x79>
  40124d: e8 2e fe ff ff          call   401080 <rand@plt>
  401252: 89 c1                   mov    %eax,%ecx
  401254: 8b 45 fc                mov    -0x4(%rbp),%eax
  401257: 48 98                   cltq
  401259: 48 8d 14 85 00 00 00    lea    0x0(,%rax,4),%rdx
  401260: 00
  401261: 48 8b 45 f0             mov    -0x10(%rbp),%rax
  401265: 48 8d 34 02             lea    (%rdx,%rax,1),%rsi
  401269: 48 63 c1                movslq %ecx,%rax
  40126c: 48 69 c0 67 66 66 66    imul   $0x66666667,%rax,%rax
```

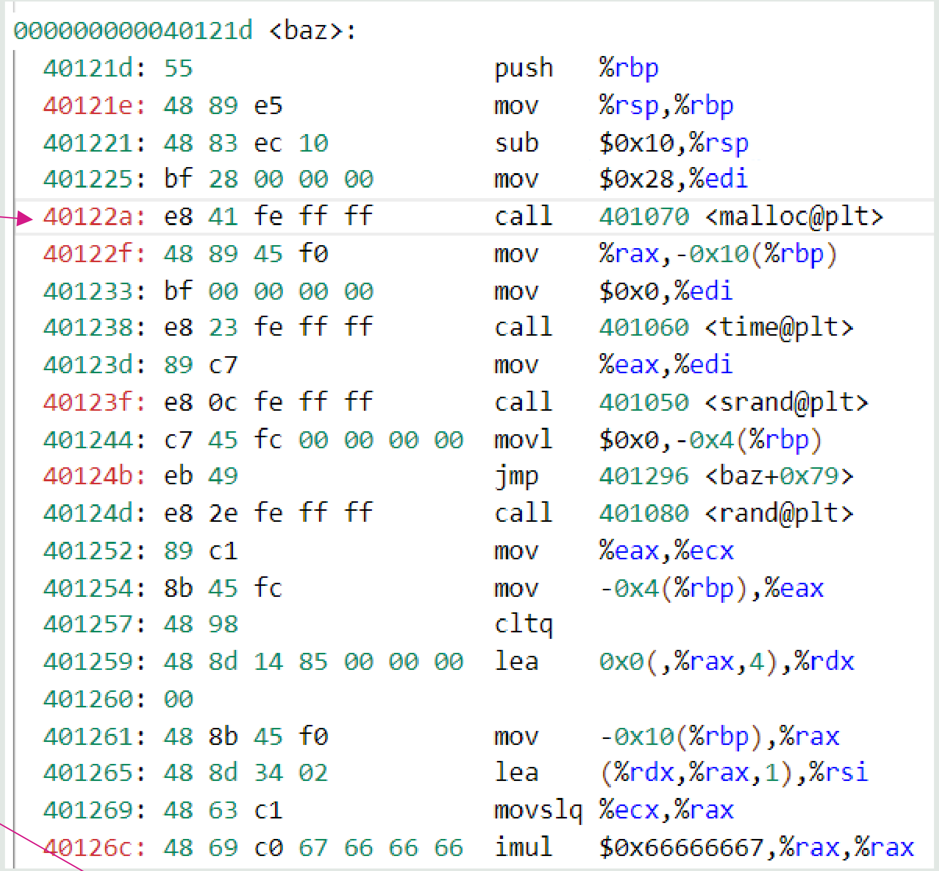
heap

- Dynamically allocated variables will be in the heap
- For e.g.



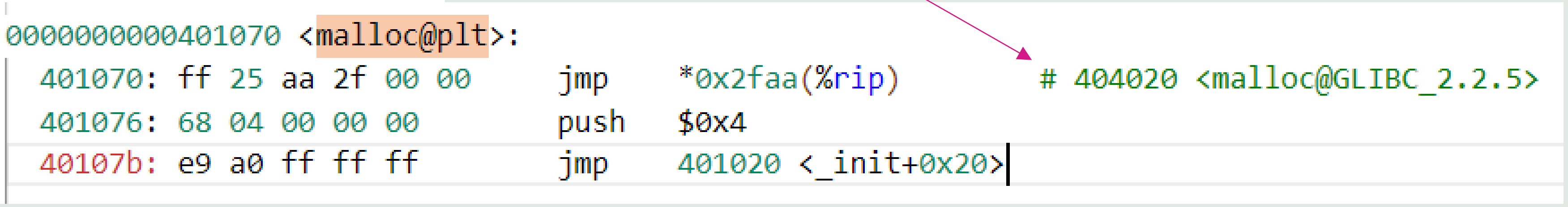
```c
void baz()
{
    // array of size 10 using malloc, so will be allocated on the heap
    int *arr = (int *)malloc(10 * sizeof(int));

    srand(time(NULL)); // seed the random number generator

    // Initialize the array randomly
    for (int i = 0; i < 10; i++)
    {
        arr[i] = rand() % 10;
    }
    printf("Elements of arr: ");
    for (int i = 0; i < 10; i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
```

Call to malloc@plt



```
000000000040121d <baz>:
  40121d: 55                    push   %rbp
  40121e: 48 89 e5              mov    %rsp,%rbp
  401221: 48 83 ec 10           sub    $0x10,%rsp
  401225: bf 28 00 00 00        mov    $0x28,%edi
  40122a: e8 41 fe ff ff        call   401070 <malloc@plt>
  40122f: 48 89 45 f0           mov    %rax,-0x10(%rbp)
  401233: bf 00 00 00 00        mov    $0x0,%edi
  401238: e8 23 fe ff ff        call   401060 <time@plt>
  40123d: 89 c7                 mov    %eax,%edi
  40123f: e8 0c fe ff ff        call   401050 <srand@plt>
  401244: c7 45 fc 00 00 00 00  movl   $0x0,-0x4(%rbp)
  40124b: eb 49                 jmp    401296 <baz+0x79>
  40124d: e8 2e fe ff ff        call   401080 <rand@plt>
  401252: 89 c1                 mov    %eax,%ecx
  401254: 8b 45 fc              mov    -0x4(%rbp),%eax
  401257: 48 98                 cltq
  401259: 48 8d 14 85 00 00 00  lea    0x0(,%rax,4),%rdx
  401260: 00
  401261: 48 8b 45 f0           mov    -0x10(%rbp),%rax
  401265: 48 8d 34 02           lea    (%rdx,%rax,1),%rsi
  401269: 48 63 c1              movslq %ecx,%rax
  40126c: 48 69 c0 67 66 66 66  imul   $0x66666667,%rax,%rax
```

Jump to GLIBC



```
0000000000401070 <malloc@plt>:
  401070: ff 25 aa 2f 00 00     jmp    *0x2faa(%rip)        # 404020 <malloc@GLIBC_2.2.5>
  401076: 68 04 00 00 00        push   $0x4
  40107b: e9 a0 ff ff ff        jmp    401020 <_init+0x20>
```

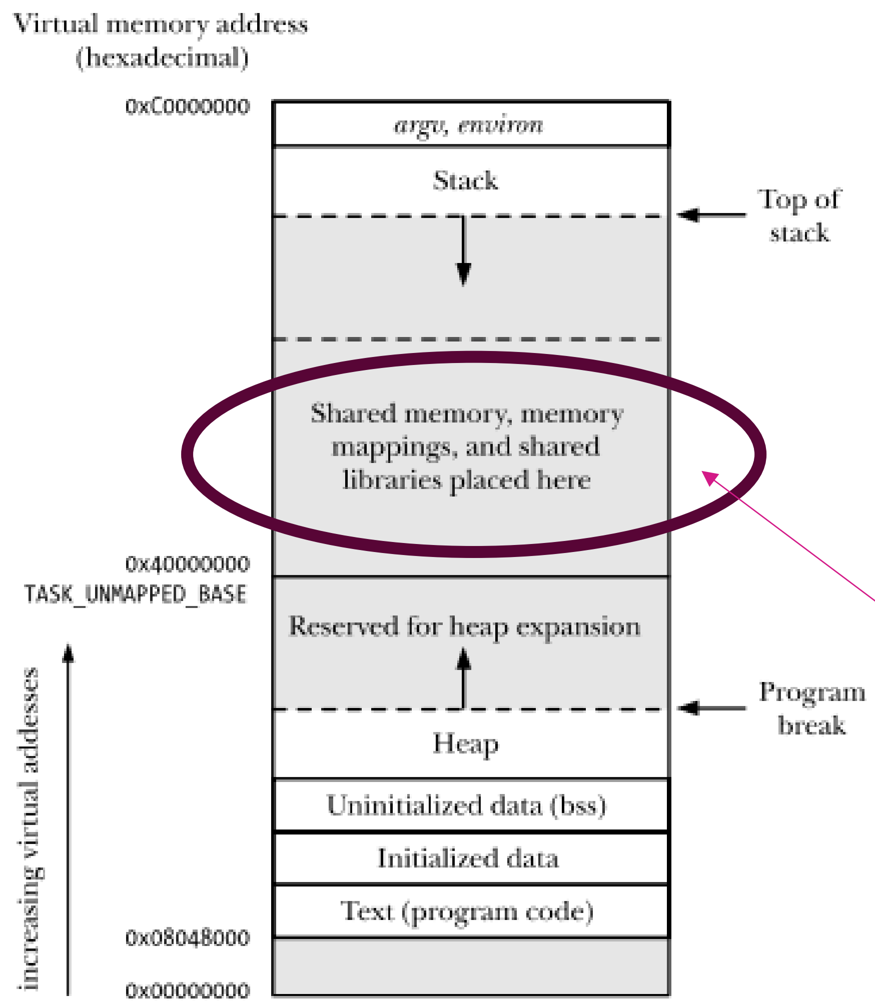


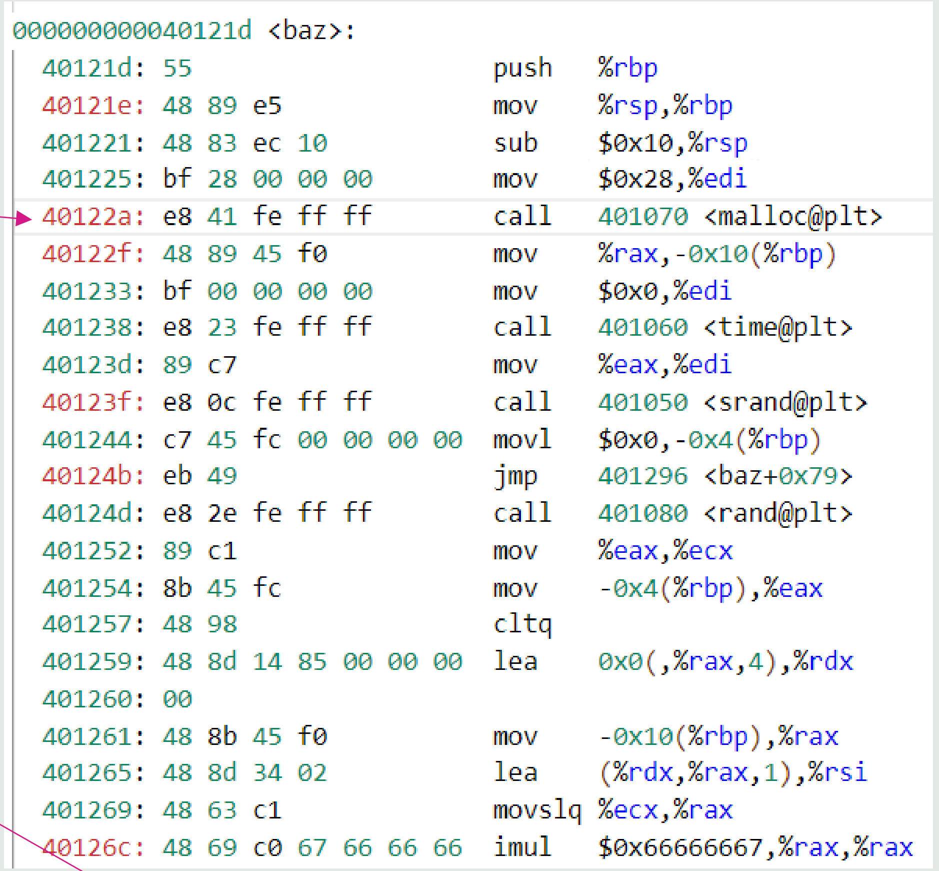
Figure 48-2: Locations of shared memory, memory mappings, and shared libraries (x86-32)

malloc@plt

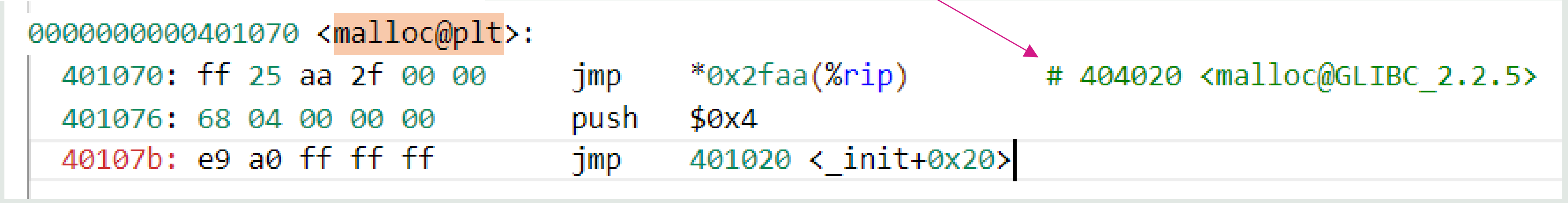
e in the heap

To be handled by linker and loader

Jump to GLIBC



```
000000000040121d <baz>:
  40121d: 55                      push   %rbp
  40121e: 48 89 e5                mov    %rsp,%rbp
  401221: 48 83 ec 10             sub    $0x10,%rsp
  401225: bf 28 00 00 00          mov    $0x28,%edi
  40122a: e8 41 fe ff ff          call   401070 <malloc@plt>
  40122f: 48 89 45 f0             mov    %rax,-0x10(%rbp)
  401233: bf 00 00 00 00          mov    $0x0,%edi
  401238: e8 23 fe ff ff          call   401060 <time@plt>
  40123d: 89 c7                   mov    %eax,%edi
  40123f: e8 0c fe ff ff          call   401050 <srand@plt>
  401244: c7 45 fc 00 00 00 00    movl   $0x0,-0x4(%rbp)
  40124b: eb 49                   jmp    401296 <baz+0x79>
  40124d: e8 2e fe ff ff          call   401080 <rand@plt>
  401252: 89 c1                   mov    %eax,%ecx
  401254: 8b 45 fc                mov    -0x4(%rbp),%eax
  401257: 48 98                   cltq
  401259: 48 8d 14 85 00 00 00    lea    0x0(,%rax,4),%rdx
  401260: 00
  401261: 48 8b 45 f0             mov    -0x10(%rbp),%rax
  401265: 48 8d 34 02             lea    (%rdx,%rax,1),%rsi
  401269: 48 63 c1                movslq %ecx,%rax
  40126c: 48 69 c0 67 66 66 66    imul   $0x66666667,%rax,%rax
```



```
0000000000401070 <malloc@plt>:
  401070: ff 25 aa 2f 00 00       jmp    *0x2faa(%rip)        # 404020 <malloc@GLIBC_2.2.5>
  401076: 68 04 00 00 00          push   $0x4
  40107b: e9 a0 ff ff ff          jmp    401020 <_init+0x20>
```