



"From C To Assembly" Cheat Sheet

Author: oa2013.github.io

Date: July 25, 2021

Sources used:

- Dennis Yurichev's [Reverse Engineering For Beginners](#)
- [Compiler Explorer](#) (x86-64 gcc 5.1)
- [Beginners Book \(C Tutorials\)](#)
- [Geeks for Geeks \(C Tutorials\)](#)

printf() with arguments

```
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("a=%d; b=%d; c=%d", 1, 2, 3);
6      return 0;
7  }
```

```
1  .LC0:
2      .string "a=%d; b=%d; c=%d"
3  main:
4      push    rbp
5      mov     rbp, rsp
6      mov     ecx, 3
7      mov     edx, 2
8      mov     esi, 1
9      mov     edi, OFFSET FLAT:.LC0
10     mov     eax, 0
11     call    printf
12     mov     eax, 0
13     pop     rbp
14     ret
15
```

scanf()

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int x;
6      scanf ("%d", &x);
7      return 0;
8  }
```

```
1  .LC0:
2      .string "%d"
3  main:
4      push    rbp
5      mov     rbp, rsp
6      sub     rsp, 16
7      lea     rax, [rbp-4]
8      mov     rsi, rax
9      mov     edi, OFFSET FLAT:.LC0
10     mov     eax, 0
11     call    __isoc99_scanf
12     mov     eax, 0
13     leave
14     ret
```

Passed arguments

```
1  #include <stdio.h>
2
3  int f(int a, int b, int c)
4  {
5      return a*b+c;
6  }
7
8  int main()
9  {
10     printf ("%d\n", f(1, 2, 3));
11     return 0;
12 }
```

```

1  f:
2      push    rbp
3      mov     rbp, rsp
4      mov     DWORD PTR [rbp-4], edi
5      mov     DWORD PTR [rbp-8], esi
6      mov     DWORD PTR [rbp-12], edx
7      mov     eax, DWORD PTR [rbp-4]
8      imul    eax, DWORD PTR [rbp-8]
9      mov     edx, eax
10     mov     eax, DWORD PTR [rbp-12]
11     add     eax, edx
12     pop     rbp
13     ret
14     .LC0:
15     .string "%d\n"
16     main:
17     push    rbp
18     mov     rbp, rsp
19     mov     edx, 3
20     mov     esi, 2
21     mov     edi, 1
22     call    f
23     mov     esi, eax
24     mov     edi, OFFSET FLAT:.LC0
25     mov     eax, 0
26     call    printf
27     mov     eax, 0
28     pop     rbp
29     ret

```

Pointers

```
1  #include <stdio.h>
2
3  int main()
4  {
5      /* Pointer p */
6      int *p;
7
8      int var = 5;
9
10     /* Assigning the address of var to p */
11     p = &var;
12
13     return 0;
14 }
```

```
1  main:
2      push    rbp
3      mov     rbp, rsp
4      mov     DWORD PTR [rbp-12], 5
5      lea     rax, [rbp-12]
6      mov     QWORD PTR [rbp-8], rax
7      mov     eax, 0
8      pop     rbp
9      ret
10
```

Double pointers

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int var = 3;
6
7      //pointer for var
8      int *ptr2;
9
10     //double pointer for ptr2
11     int **ptr1;
12
13     //storing address of var in ptr2
14     ptr2 = &var;
15
16     //storing address of ptr2 in ptr1
17     ptr1 = &ptr2;
18
19     return 0;
20 }
```

```
1  main:
2      push    rbp
3      mov     rbp, rsp
4      mov     DWORD PTR [rbp-12], 3
5      lea     rax, [rbp-12]
6      mov     QWORD PTR [rbp-24], rax
7      lea     rax, [rbp-24]
8      mov     QWORD PTR [rbp-8], rax
9      mov     eax, 0
10     pop     rbp
11     ret
12
```

Arrays

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int arr[5] = {1, 2, 3, 4, 5};
6      return 0;
7  }
```

```
1  main:
2      push    rbp
3      mov     rbp, rsp
4      mov     DWORD PTR [rbp-32], 1
5      mov     DWORD PTR [rbp-28], 2
6      mov     DWORD PTR [rbp-24], 3
7      mov     DWORD PTR [rbp-20], 4
8      mov     DWORD PTR [rbp-16], 5
9      mov     eax, 0
10     pop     rbp
11     ret
```

For loop

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int i;
6
7      for (i=1; i<=4; i++)
8      {
9          printf("%d\n", i);
10     }
11
12     return 0;
13 }
```

```

1  .LC0:
2      .string "%d\n"
3  main:
4      push    rbp
5      mov     rbp, rsp
6      sub     rsp, 16
7      mov     DWORD PTR [rbp-4], 1
8      jmp     .L2
9  .L3:
10     mov     eax, DWORD PTR [rbp-4]
11     mov     esi, eax
12     mov     edi, OFFSET FLAT:.LC0
13     mov     eax, 0
14     call    printf
15     add     DWORD PTR [rbp-4], 1
16  .L2:
17     cmp     DWORD PTR [rbp-4], 4
18     jle     .L3
19     mov     eax, 0
20     leave
21     ret

```

If...else

```

1  #include <stdio.h>
2
3  int main()
4  {
5      int salary;
6
7      salary = 17;
8
9      if(salary >=15)
10     {
11         printf("Above min wage");
12     }
13     else
14     {
15         printf("Below min wage");
16     }
17
18     return 0;
19 }

```

```

1  .LC0:
2      .string "Above min wage"
3  .LC1:
4      .string "Below min wage"
5  main:
6      push    rbp
7      mov     rbp, rsp
8      sub     rsp, 16
9      mov     DWORD PTR [rbp-4], 17
10     cmp     DWORD PTR [rbp-4], 14
11     jle     .L2
12     mov     edi, OFFSET FLAT:.LC0
13     mov     eax, 0
14     call    printf
15     jmp     .L3
16 .L2:
17     mov     edi, OFFSET FLAT:.LC1
18     mov     eax, 0
19     call    printf
20 .L3:
21     mov     eax, 0
22     leave
23     ret

```

While loop

```

1  #include <stdio.h>
2
3  int main()
4  {
5      int count=1;
6
7      while (count <= 3)
8      {
9          printf("%d ", count);
10         count++;
11     }
12
13     return 0;
14 }

```



```
1  .LC0:
2      .string "%d "
3  main:
4      push    rbp
5      mov     rbp, rsp
6      sub     rsp, 16
7      mov     DWORD PTR [rbp-4], 1
8      jmp     .L2
9  .L3:
10     mov     eax, DWORD PTR [rbp-4]
11     mov     esi, eax
12     mov     edi, OFFSET FLAT:.LC0
13     mov     eax, 0
14     call    printf
15     add     DWORD PTR [rbp-4], 1
16  .L2:
17     cmp     DWORD PTR [rbp-4], 3
18     jle     .L3
19     mov     eax, 0
20     leave
21     ret
```

Structures

```
1  #include <stdio.h>
2
3  struct Food{
4      char *name;
5      int cost;
6      int foodId;
7  };
8
9  int main()
10 {
11     struct Food orange;
12
13     orange.name = "Blood Orange";
14     orange.cost = 3;
15     orange.foodId = 1;
16
17     printf("Food name is: %s \n", orange.name);
18     printf("Food cost is: %d \n", orange.cost);
19     printf("Food id is: %d \n", orange.foodId);
20
21     return 0;
22 }
```

```

1  .LC0:
2      .string "Blood Orange"
3  .LC1:
4      .string "Food name is: %s \n"
5  .LC2:
6      .string "Food cost is: %d \n"
7  .LC3:
8      .string "Food id is: %d \n"
9  main:
10     push    rbp
11     mov     rbp, rsp
12     sub     rsp, 16
13     mov     QWORD PTR [rbp-16], OFFSET FLAT:.LC0
14     mov     DWORD PTR [rbp-8], 3
15     mov     DWORD PTR [rbp-4], 1
16     mov     rax, QWORD PTR [rbp-16]
17     mov     rsi, rax
18     mov     edi, OFFSET FLAT:.LC1
19     mov     eax, 0
20     call    printf
21     mov     eax, DWORD PTR [rbp-8]
22     mov     esi, eax
23     mov     edi, OFFSET FLAT:.LC2
24     mov     eax, 0
25     call    printf
26     mov     eax, DWORD PTR [rbp-4]
27     mov     esi, eax
28     mov     edi, OFFSET FLAT:.LC3
29     mov     eax, 0
30     call    printf
31     mov     eax, 0
32     leave
33     ret

```