P1.a

Friday, March 10, 2023 2:21 AM

IEEE-754:

Sign: it's negative: ∴ sign is 1

55 in Binary: 110111 0.375 in Binary: .0110

∴ 55.375 = 110111.0110000... ← this should move back 5 spaces.

$= 1.10111011000000... \times 2^5$ ← this is the exponent, $5+127 = 132$.

∴ Exponent = 132 in Binary = 10000100

Mantissa = 10111011000000.... ← get rid of the first 1.

∴ Float Point Rep in Binary is 1100 0010 0101 1101 1000 0000 ...

C 2 5 D 8 0 0 0

in Hex Rep: 0x C25D8000

P1.b

Friday, March 10, 2023 2:56 AM

b): 0_x 404 cccc d. in Binary is.

4	0	4	c	c	c	c	d
0100	0000	0100	1100	1100	1100	1100	1101

Sign: 0 Exp: 0100 0000 Matissa: 0100 1100 1100 1100 1100 1101

sign: it's positive.

Exp = 128 , $128 - 127 = 1 \Rightarrow 2^1$

Matissa = 1.10011100 1100 1100 1100 1101 ← move 1 to the right.

= 11.0011 0011 0011 0011

= 3.33333....

∴ the real number is $10/3$.

P1.cd

Friday, March 10, 2023 3:06 AM

c) :

	Exp	Mantissa
NaN :	255	Non-Zero

∴ Sign : x

Exp : 255 = 11111111

Man : xxxx xxxx ..... (non-zero)

∴ NaN : x111 1111 1xxx xxxx xxxx xxxx ....

if x all equal 0, except for this one, this one equals to 1.

NaN : 0111 1111 1100 0000 0000 0000 ....

7 F C 0 0 0 0 0

NaN in Hex : 0x7FC00000

d) : Found this online. https://www.geeksforgeeks.org/ieee-standard-754-floating-point-numbers/

```
int main()
{
    float x = 0.0 / 0.0;
    printf("%f\n", x);

    return 0;
}
```

P1.ef

Friday, March 10, 2023 3:31 AM

e):

	Sign	Exp	Mantissa
-inf	1	255	0

⇒ sign = 1

Exp = 255 = 1111111

Mantissa = 0000 0000 0000 ----

⇒ -inf = 1111 1111 1000 0000 0000 ----

F F 8 0 0

-inf in Hex is 0x FF800000

f): Found this online. → https://www.geeksforgeeks.org/ieee-standard-754-floating-point-numbers/

```
int main()
{
    float x = -5.0 / 0.0;
    printf("%f\n", x);

    return 0;
}
```

P2

Sunday, March 12, 2023 8:17 PM

```c
#include <stdio.h>

int do_something(int a, int b, int c, int d, int e, int f, int g)
{
    return a + b + c + d + e + f + g;
}

int main()
{
    int a = 1;
    int b = 2;
    int c = 3;
    int d = 4;
    int e = 5;
    int f = 6;
    int g = 7;

    printf("%d\n", do_something(a, b, c, d, e, f, g));

    return 0;
}
```

I complied the codes here with "gcc -Wall -Werror -g -O0 -no-pie -fno-stack-protector -o lotsafuncs lotsafuncs.c"

And run [lotsafunc] with [gdb]. And [objdump] the page to a text file: [objdump]

In [objdump] file we can see that when preparing the registers to call [do_something] function, the (main) function made use of the [rbp] register. This register is set to point at where the stack pointer was originally pointing at, so that when we move the stack pointer around we can still have access to the data referenced by [rbp] by using relative addresses.

In this case, the parameters for [do_something] function are stored in the stack, and is referenced by the block pointer [rbp]

I suspect that moving the values referenced by [rbp] to free registers in line 4011b2 to 4011c2 and moving them back (line 40113e to 40114e) is caused by bad compiler optimization.

```
0000000000401175 <main>:
  401175:       f3 0f 1e fa             endbr64
  401179:       55                      push   %rbp
  40117a:       48 89 e5                mov    %rsp,%rbp
  40117d:       48 83 ec 20             sub    $0x20,%rsp
  401181:       c7 45 fc 01 00 00 00    movl   $0x1,-0x4(%rbp)
  401188:       c7 45 f8 02 00 00 00    movl   $0x2,-0x8(%rbp)
  40118f:       c7 45 f4 03 00 00 00    movl   $0x3,-0xc(%rbp)
  401196:       c7 45 f0 04 00 00 00    movl   $0x4,-0x10(%rbp)
  40119d:       c7 45 ec 05 00 00 00    movl   $0x5,-0x14(%rbp)
  4011a4:       c7 45 e8 06 00 00 00    movl   $0x6,-0x18(%rbp)
  4011ab:       c7 45 e4 07 00 00 00    movl   $0x7,-0x1c(%rbp)
  4011b2:       44 8b 4d e8             mov    -0x18(%rbp),%r9d
  4011b6:       44 8b 45 ec             mov    -0x14(%rbp),%r8d
  4011ba:       8b 4d f0                mov    -0x10(%rbp),%ecx
  4011bd:       8b 55 f4                mov    -0xc(%rbp),%edx
  4011c0:       8b 75 f8                mov    -0x8(%rbp),%esi
  4011c3:       8b 45 fc                mov    -0x4(%rbp),%eax
  4011c6:       8b 7d e4                mov    -0x1c(%rbp),%edi
  4011c9:       57                      push   %rdi
  4011ca:       89 c7                   mov    %eax,%edi
  4011cc:       e8 65 ff ff ff          callq  401136 <do_something>
```

```
0000000000401136 <do_something>:
  401136:       f3 0f 1e fa             endbr64
  40113a:       55                      push   %rbp
  40113b:       48 89 e5                mov    %rsp,%rbp
  40113e:       89 7d fc                mov    %edi,-0x4(%rbp)
  401141:       89 75 f8                mov    %esi,-0x8(%rbp)
  401144:       89 55 f4                mov    %edx,-0xc(%rbp)
  401147:       89 4d f0                mov    %ecx,-0x10(%rbp)
  40114a:       44 89 45 ec             mov    %r8d,-0x14(%rbp)
  40114e:       44 89 4d e8             mov    %r9d,-0x18(%rbp)
  401152:       8b 55 fc                mov    -0x4(%rbp),%edx
  401155:       8b 45 f8                mov    -0x8(%rbp),%eax
  401158:       01 c2                   add    %eax,%edx
  40115a:       8b 45 f4                mov    -0xc(%rbp),%eax
  40115d:       01 c2                   add    %eax,%edx
  40115f:       8b 45 f0                mov    -0x10(%rbp),%eax
  401162:       01 c2                   add    %eax,%edx
  401164:       8b 45 ec                mov    -0x14(%rbp),%eax
  401167:       01 c2                   add    %eax,%edx
  401169:       8b 45 e8                mov    -0x18(%rbp),%eax
  40116c:       01 c2                   add    %eax,%edx
  40116e:       8b 45 10                mov    0x10(%rbp),%eax
  401171:       01 d0                   add    %edx,%eax
  401173:       5d                      pop    %rbp
  401174:       c3                      retq
```

P3.a1

Monday, March 13, 2023 3:32 PM

```
0x7fffffffe3b0: 0x00000980      0x00000980      0x00000980      0x00000001
0x7fffffffe3c0: 0xffffe3e0      0x00007fff      0x0040115f      0x00000000
0x7fffffffe3d0: 0x00000000      0x00000040      0x00000200      0x00000002
0x7fffffffe3e0: 0xffffe400      0x00007fff      0x0040115f      0x00000000
0x7fffffffe3f0: 0x00000000      0x00000000      0x00000000      0x00000003
0x7fffffffe400: 0xffffe420      0x00007fff      0x0040115f      0x00000000
0x7fffffffe410: 0x00000000      0x00000000      0x00000000      0x00000004
0x7fffffffe420: 0xffffe440      0x00007fff      0x0040115f      0x00000000
0x7fffffffe430: 0x000000c2      0x00000000      0xffffe467      0x00000005
0x7fffffffe440: 0xffffe460      0x00007fff      0x0040115f      0x00000000
0x7fffffffe450: 0xf7fbcfc8      0x00007fff      0x004011a0      0x00000006
0x7fffffffe460: 0xffffe480      0x00007fff      0x0040117c      0x00000000
0x7fffffffe470: 0xffffe570      0x00007fff      0x00000000      0x00000000
0x7fffffffe480: 0x00000000      0x00000000      0xf7df30b3      0x00007fff
0x7fffffffe490: 0xf7ffc620      0x00007fff      0xffffe578      0x00007fff
```

40114b: we put 1 in edx, which corresponds to this number.

401164: leave q (move rbp to rsp, pop rbp), this moves the stack pointer (rsp) to what rbp was pointing: 0x7fffffffe3c0 and pop the value stored there to rbp, so at this point rbp has the value 0x7fffffffe3e0. Also, bc we've performed a pop, rsp moved to e3c8.

401165: return q, pop the first 8 bytes of rsp jump to that address. 40115f was popped, now we're at line 40115f and rsp is pointing at e3d0

```
  0x401136 <triangular_number>          endbr64
  0x40113a <triangular_number+4>        push   %rbp
  0x40113b <triangular_number+5>        mov    %rsp,%rbp
  0x40113e <triangular_number+8>        sub    $0x10,%rsp
  0x401142 <triangular_number+12>       mov    %edi,-0x4(%rbp)
  0x401145 <triangular_number+15>       cmpl   $0x1,-0x4(%rbp)
  0x401149 <triangular_number+19>       jg     0x401152 <triangular_number+28>
+ 0x40114b <triangular_number+21>       mov    $0x1,%eax
  0x401150 <triangular_number+26>       jmp    0x401164 <triangular_number+46>
  0x401152 <triangular_number+28>       mov    -0x4(%rbp),%eax
  0x401155 <triangular_number+31>       sub    $0x1,%eax
  0x401158 <triangular_number+34>       mov    %eax,%edi
  0x40115a <triangular_number+36>       callq  0x401136 <triangular_number>
  0x40115f <triangular_number+41>       mov    -0x4(%rbp),%edx
  0x401162 <triangular_number+44>       add    %edx,%eax
>0x401164 <triangular_number+46>        leaveq
  0x401165 <triangular_number+47>       retq
```

P3.a2

Monday, March 13, 2023 9:20 PM

```
0x7fffffffe3d0: 0x00000000      0x00000040      0x00000200      0x00000002
0x7fffffffe3e0: 0xffffe400      0x00007fff      0x0040115f      0x00000000
0x7fffffffe3f0: 0x00000000      0x00000000      0x00000000      0x00000003
0x7fffffffe400: 0xffffe420      0x00007fff      0x0040115f      0x00000000
0x7fffffffe410: 0x00000000      0x00000000      0x00000000      0x00000004
0x7fffffffe420: 0xffffe440      0x00007fff      0x0040115f      0x00000000
0x7fffffffe430: 0x000000c2      0x00000000      0xffffe467      0x00000005
0x7fffffffe440: 0xffffe460      0x00007fff      0x0040115f      0x00000000
0x7fffffffe450: 0xf7fbcfc8      0x00007fff      0x004011a0      0x00000006
0x7fffffffe460: 0xffffe480      0x00007fff      0x0040117c      0x00000000
0x7fffffffe470: 0xffffe570      0x00007fff      0x00000000      0x00000000
0x7fffffffe480: 0x00000000      0x00000000      0xf7df30b3      0x00007fff
0x7fffffffe490: 0xf7ffc620      0x00007fff      0xffffe578      0x00007fff
0x7fffffffe4a0: 0x00000000      0x00000001      0x00401166      0x00000000
0x7fffffffe4b0: 0x004011a0      0x00000000      0x2cb7628b      0xd4211c82
```

40115f: move the number stored in ($rbp - 4) into edx. Rbp is pointing at e3e0, so ($rbp - 4) is e3dc, where the number 2 is stored. So edx = 2.

401162: add edx to eax. Eax is the return value from the last call of <triangular_number>, it has a value of 1. so now eax = 1 + 2 = 3, and it will be the return value of the 4th call of <triangular_number>

401164: leave q (move rbp to rsp, pop rbp), this moves the stack pointer (rsp) to what rbp was pointing: e3e0 and pop the value stored there to rbp, so at this point rbp has the value e400. Also, bc we've performed a pop, rsp is moved to e3e8.

401165: return q, pop the first 8 bytes of rsp jump to that address. 40115f was popped, now we're back at line 40115f and rsp is pointing at e3f0

```
 0x401136 <triangular_number>          endbr64
 0x40113a <triangular_number+4>        push   %rbp
 0x40113b <triangular_number+5>        mov    %rsp,%rbp
 0x40113e <triangular_number+8>        sub    $0x10,%rsp
 0x401142 <triangular_number+12>       mov    %edi,-0x4(%rbp)
 0x401145 <triangular_number+15>       cmpl   $0x1,-0x4(%rbp)
 0x401149 <triangular_number+19>       jg     0x401152 <triangular_number+28>
 0x40114b <triangular_number+21>       mov    $0x1,%eax
 0x401150 <triangular_number+26>       jmp    0x401164 <triangular_number+46>
 0x401152 <triangular_number+28>       mov    -0x4(%rbp),%eax
 0x401155 <triangular_number+31>       sub    $0x1,%eax
 0x401158 <triangular_number+34>       mov    %eax,%edi
 0x40115a <triangular_number+36>       callq  0x401136 <triangular_number>
>0x40115f <triangular_number+41>       mov    -0x4(%rbp),%edx
 0x401162 <triangular_number+44>       add    %edx,%eax
 0x401164 <triangular_number+46>       leaveq
 0x401165 <triangular_number+47>       retq
```

P3.a3

Monday, March 13, 2023 9:32 PM

We repeat the process 3 more time until we hit the first call of <triangular_number> .

Now the stack frame looks like this:

```
0x7fffffffe450: 0xf7fbcfc8      0x00007fff      0x004011a0      0x00000006
0x7fffffffe460: 0xffffe480      0x00007fff      0x0040117c      0x00000000
0x7fffffffe470: 0xffffe570      0x00007fff      0x00000000      0x00000000
```

40115f: move 6 into edx.

401162: add edx to eax. Now eax = 22 + 6 = 28. this will be the return value.

401164: leave q (move rbp to rsp, pop rbp), this moves the stack pointer (rsp) to what rbp was pointing: e460 and pop the value stored there to rbp, so at this point rbp has the value e480. Also, bc we've performed a pop, rsp is moved to e468.

401165: return q, pop the first 8 bytes of rsp jump to that address. 40117c was popped, now we're back in <main> at line 40117c. With the return value eax = 28.

```
 0x401136 <triangular_number>    endbr64
 0x40113a <triangular_number+4>  push   %rbp
 0x40113b <triangular_number+5>  mov    %rsp,%rbp
 0x40113e <triangular_number+8>  sub    $0x10,%rsp
 0x401142 <triangular_number+12> mov    %edi,-0x4(%rbp)
 0x401145 <triangular_number+15> cmpl   $0x1,-0x4(%rbp)
 0x401149 <triangular_number+19> jg     0x401152 <triangular_number+28>
 0x40114b <triangular_number+21> mov    $0x1,%eax
 0x401150 <triangular_number+26> jmp    0x401164 <triangular_number+46>
 0x401152 <triangular_number+28> mov    -0x4(%rbp),%eax
 0x401155 <triangular_number+31> sub    $0x1,%eax
 0x401158 <triangular_number+34> mov    %eax,%edi
 0x40115a <triangular_number+36> callq  0x401136 <triangular_number>
>0x40115f <triangular_number+41> mov    -0x4(%rbp),%edx
 0x401162 <triangular_number+44> add    %edx,%eax
 0x401164 <triangular_number+46> leaveq
 0x401165 <triangular_number+47> retq
```

P3.b1

Monday, March 13, 2023 11:39 AM

```
0x401166 <main>          endbr64
0x40116a <main+4>        push   %rbp
0x40116b <main+5>        mov    %rsp,%rbp
0x40116e <main+8>        sub    $0x10,%rsp
0x401172 <main+12>       mov    $0x6,%edi
0x401177 <main+17>       callq  0x401136 <triangular_number>
0x40117c <main+22>       mov    %eax,-0x4(%rbp)
0x40117f <main+25>       mov    -0x4(%rbp),%eax
0x401182 <main+28>       mov    %eax,%esi
0x401184 <main+30>       lea    0xe79(%rip),%rdi        # 0x402004
0x40118b <main+37>       mov    $0x0,%eax
0x401190 <main+42>       callq  0x401040 <printf@plt>
0x401195 <main+47>       mov    $0x0,%eax
0x40119a <main+52>       leaveq
0x40119b <main+53>       retq
```

401166-401172: set up the parameters for <triangular_number>. At line 40116a rbp is 0 and at line 40116b the current address stored in rsp (e488) is copied into rbp and is stored on the stack. Then we subtract 10 from rsp and set rbp to 6.

401177: call <triabgular_number>.

401166: 40117c is the return address. The stack looks like this:

```
0x7fffffffe468: 0x0040117c      0x00000000      0xffffe570      0x00007fff
0x7fffffffe478: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffffffe488: 0xf7df30b3      0x00007fff      0xf7ffc620      0x00007fff
0x7fffffffe498: 0xffffe578      0x00007fff      0x00000000      0x00000001
0x7fffffffe4a8: 0x00401166      0x00000000      0x004011a0      0x00000000
```

40116a: push rbp onto the stack: E480 is the value of rbp, it represents the stack pointer address in <main>

```
0x7fffffffe460: 0xffffe480      0x00007fff      0x0040117c      0x00000000
0x7fffffffe470: 0xffffe570      0x00007fff      0x00000000      0x00000000
0x7fffffffe480: 0x00000000      0x00000000      0xf7df30b3      0x00007fff
0x7fffffffe490: 0xf7ffc620      0x00007fff      0xffffe578      0x00007fff
0x7fffffffe4a0: 0x00000000      0x00000001      0x00401166      0x00000000
```

40113e: subtract 10 from rsp. Allocate 10 bytes in the stack. Now rsp is e450.

```
0x7fffffffe450: 0xf7fbcfc8      0x00007fff      0x004011a0      0x00000000
0x7fffffffe460: 0xffffe480      0x00007fff      0x0040117c      0x00000000
0x7fffffffe470: 0xffffe570      0x00007fff      0x00000000      0x00000000
0x7fffffffe480: 0x00000000      0x00000000      0xf7df30b3      0x00007fff
0x7fffffffe490: 0xf7ffc620      0x00007fff      0xffffe578      0x00007fff
```

401142: move edi (6) into ($rbp - 4). Bc the address of rbp is e460, so ($rbp - 4) = e45c

this 6 is moved here from edi

```
0x7fffffffe450: 0xf7fbcfc8      0x00007fff      0x004011a0      0x00000006
0x7fffffffe460: 0xffffe480      0x00007fff      0x0040117c      0x00000000
0x7fffffffe470: 0xffffe570      0x00007fff      0x00000000      0x00000000
0x7fffffffe480: 0x00000000      0x00000000      0xf7df30b3      0x00007fff
0x7fffffffe490: 0xf7ffc620      0x00007fff      0xffffe578      0x00007fff
```

401145: compare the number stored at ($rbp - 4) with 1. if greater jump to <triangular_number>+28. the value at ($rbp - 4) is 6 so we jump.

401152 - 401158 (<triangular_number> 28 - 34) : put the value in ($rbp - 4) to eax (eax = 6), subtract 1 from eax (eax = 5), put eax in edi (edi = 5). This whole process sets up the parameter for the second <triangular_number> call (edi = 5).

P3.b2

Monday, March 13, 2023 9:40 PM

The second call of <triangular_number>

40113a: push rbp. Stored the address of rbp from the first call of <triangular_number> (e460), this is used to reference the previous stack pointer address.

```
0x7fffffffe440: 0xffffe460      0x00007fff      0x0040115f      0x00000000
0x7fffffffe450: 0xf7fbcfc8      0x00007fff      0x004011a0      0x00000006
0x7fffffffe460: 0xffffe480      0x00007fff      0x0040117c      0x00000000
0x7fffffffe470: 0xffffe570      0x00007fff      0x00000000      0x00000000
0x7fffffffe480: 0x00000000      0x00000000      0xf7df30b3      0x00007fff
```

40113b - 40113e: copy the current stack pointer address into rbp (rbp = e440). And allocate 10 bytes in the stack (subtract 10 from the stack pointer). Now rsp is e430.

401142: move edi (5) to ($rbp - 4). Bc the address of rbp is e440, ($rbp - 4) = e43c. We put the number 5 there.

```
0x7fffffffe430: 0x000000c2      0x00000000      0xffffe467      0x00000005
0x7fffffffe440: 0xffffe460      0x00007fff      0x0040115f      0x00000000
0x7fffffffe450: 0xf7fbcfc8      0x00007fff      0x004011a0      0x00000006
0x7fffffffe460: 0xffffe480      0x00007fff      0x0040117c      0x00000000
0x7fffffffe470: 0xffffe570      0x00007fff      0x00000000      0x00000000
```

here

The rest is similar to P3.b1

This process goes on until we reached a point when edi = 1, which is where we make our last call of <triangular_number>:

Last call of <triangular_number>:
401142: stack frame looks like this:

```
0x7fffffffe3b0: 0x00000980      0x00000980      0x00000980      0x00000001
0x7fffffffe3c0: 0xffffe3e0      0x00007fff      0x0040115f      0x00000000
0x7fffffffe3d0: 0x00000000      0x00000040      0x00000200      0x00000002
0x7fffffffe3e0: 0xffffe400      0x00007fff      0x0040115f      0x00000000
0x7fffffffe3f0: 0x00000000      0x00000000      0x00000000      0x00000003
0x7fffffffe400: 0xffffe420      0x00007fff      0x0040115f      0x00000000
0x7fffffffe410: 0x00000000      0x00000000      0x00000000      0x00000004
0x7fffffffe420: 0xffffe440      0x00007fff      0x0040115f      0x00000000
0x7fffffffe430: 0x000000c2      0x00000000      0xffffe467      0x00000005
0x7fffffffe440: 0xffffe460      0x00007fff      0x0040115f      0x00000000
0x7fffffffe450: 0xf7fbcfc8      0x00007fff      0x004011a0      0x00000006
0x7fffffffe460: 0xffffe480      0x00007fff      0x0040117c      0x00000000
0x7fffffffe470: 0xffffe570      0x00007fff      0x00000000      0x00000000
0x7fffffffe480: 0x00000000      0x00000000      0xf7df30b3      0x00007fff
0x7fffffffe490: 0xf7ffc620      0x00007fff      0xffffe578      0x00007fff
```

401145: when we compare ($rbp - 4) with 1, we can see that they are equal. We proceed to line 40114b.

Now we have the stack frame set up, the rest is identical to P3.a

Some explanation of the stack frame:
Starting at e3bc: this is the parameter of the last call of <triangular_number>, we'll add it to edx and use it as the return value.
E3c0-e3c8: this is rbp in the last call of <tri-num> it is used to store the address of the stack pointer of the previous call of <tri_num>.
E3c8-e3cc: the return address of the last call of <tri_num>, when <tri_num> is finished we jump to that instruction.

P4.a

Monday, March 13, 2023 11:38 AM

This is what the stack frame looks like at the beginning of <person_test> (4012ab).

```
0x7fffffffe498: 0x0040151f      0x00000000      0x00000000      0x00000000
0x7fffffffe4a8: 0xf7df30b3      0x00007fff      0xf7ffc620      0x00007fff
0x7fffffffe4b8: 0xffffe598      0x00007fff      0x00000000      0x00000001
0x7fffffffe4c8: 0x0040150d      0x00000000      0x00401530      0x00000000
0x7fffffffe4d8: 0x542cd018      0xff80c4a2      0x00401070      0x00000000
```

The first value stored on the top of the stack (40151f) is the return address to go back to <main>.

4012ac: push rbp (e4a0) onto the stack, this is the rsp value in <main>'s stack frame. We want to store is to be able to go back to that address.

```
0x7fffffffe490: 0xffffe4a0      0x00007fff      0x0040151f      0x00000000
0x7fffffffe4a0: 0x00000000      0x00000000      0xf7df30b3      0x00007fff
0x7fffffffe4b0: 0xf7ffc620      0x00007fff      0xffffe598      0x00007fff
0x7fffffffe4c0: 0x00000000      0x00000001      0x0040150d      0x00000000
0x7fffffffe4d0: 0x00401530      0x00000000      0x542cd018      0xff80c4a2
```

4012af: push the current value of rsp to rbp (e490). We can use rbp to reference to the stack pointer of <preson_test> anytime we want.

```
0x7fffffffe490: 0xffffe4a0      0x00007fff      0x0040151f      0x00000000
0x7fffffffe4a0: 0x00000000      0x00000000      0xf7df30b3      0x00007fff
0x7fffffffe4b0: 0xf7ffc620      0x00007fff      0xffffe598      0x00007fff
0x7fffffffe4c0: 0x00000000      0x00000001      0x0040150d      0x00000000
0x7fffffffe4d0: 0x00401530      0x00000000      0x542cd018      0xff80c4a2
```

4012b0: sub 0xa8 from rsp. This is allocating 168 bytes to store the parameters for <create_preson>.

```
0x7fffffffe3e0: 0x00000000      0x00000000      0x00000100      0x00000000
0x7fffffffe3f0: 0x00000000      0x00000040      0x00000200      0x00000400
0x7fffffffe400: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffffffe410: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffffffe420: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffffffe430: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffffffe440: 0x00400040      0x00000000      0x000000f0      0x00000000
0x7fffffffe450: 0x000000c2      0x00000000      0xffffe487      0x00007fff
0x7fffffffe460: 0xffffe486      0x00007fff      0x0040157d      0x00000000
0x7fffffffe470: 0xf7fbcfc8      0x00007fff      0x00401530      0x00000000
0x7fffffffe480: 0x00000000      0x00000000      0x00401530      0x00000000
0x7fffffffe490: 0xffffe4a0      0x00007fff      0x0040151f      0x00000000
0x7fffffffe4a0: 0x00000000      0x00000000      0xf7df30b3      0x00007fff
0x7fffffffe4b0: 0xf7ffc620      0x00007fff      0xffffe598      0x00007fff
0x7fffffffe4c0: 0x00000000      0x00000001      0x0040150d      0x00000000
```

4012b7: subtract 0xb0 from rbp and put that address in rax. Rax will be used to store person_t variable. (now rax and rsp both have the value e3e0). The stack frame does not change from the previous line.

4012be - 4012c9: move the parameters that fits in a register into their registers. (23 in r8d, 6 in ecx, 1912 in edx)

Then we call <create_person> to set up our person "Alan Turing".

So when a parameter is too large to store in a register, the compiler will store it in the stack and store the address in a register to reference it.

P4.b

Wednesday, March 15, 2023 2:22 AM

This is what the stack frame looks like just after the second call of <create_person>:

```
0x7fffffffe3d8: 0x004013ad      0x00000000      0x6e616c41      0x72755420
0x7fffffffe3e8: 0x00676e69      0x00000000      0x00000000      0x00000000
0x7fffffffe3f8: 0x00000000      0x00000000      0x00000778      0x00000006
0x7fffffffe408: 0x00000017      0x00000000      0x00000000      0x00000000
0x7fffffffe418: 0x00000000      0x00000000      0x00000000      0x00000000
```

4013ad is the return address to return to when <create_person> finishes.

In <create_person>:
40115a: push rbp (e490) onto the stack so we can retrieve it later.

40115b: copy the current rsp value (e3d0) to rbp.

40115e: push rbx (0000) to the stack. (don't know why)

```
0x7fffffffe3c8: 0x00000000      0x00000000      0xffffe490      0x00007fff
0x7fffffffe3d8: 0x004013ad      0x00000000      0x6e616c41      0x72755420
0x7fffffffe3e8: 0x00676e69      0x00000000      0x00000000      0x00000000
0x7fffffffe3f8: 0x00000000      0x00000000      0x00000778      0x00000006
0x7fffffffe408: 0x00000017      0x00000000      0x00000000      0x00000000
```

40115f: subtract 88 bytes from rsp to make room to store the parameters.

401163: store rdi on an address that's 72 bytes less than rbp (e3d0) (don't know why)

```
0x7fffffffe370: 0x00000340      0x00000017      0x00000006      0x00000778
0x7fffffffe380: 0x00402004      0x00000006      0xffffe3e0      0x00007fff
0x7fffffffe390: 0x6e616c41      0x72755420      0x01676e69      0x0000006e
0x7fffffffe3a0: 0xffffe490      0x00007fff      0x00401365      0x00000000
0x7fffffffe3b0: 0x6e616c41      0x72755420      0x00676e69      0x00000000
0x7fffffffe3c0: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffffffe3d0: 0xffffe490      0x00007fff      0x004013ad      0x00000000
0x7fffffffe3e0: 0x6e616c41      0x72755420      0x00676e69      0x00000000
0x7fffffffe3f0: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffffffe400: 0x00000778      0x00000006      0x00000017      0x00000000
0x7fffffffe410: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffffffe420: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffffffe430: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffffffe440: 0x6e616c41      0x72755420      0x00676e69      0x00000000
0x7fffffffe450: 0x00000000      0x00000000      0x00000000      0x00000000
```

401167 - 401171:
store rsi (the address of name "John von Neumann", 40202a) in the address that's 90 bytes less than rbp (e3d0)
Store edx (year, 1903, 76f) in the address that's 94 bytes less than rbp (e3d0)
Store ecx (month, 12, c) in the address that's 98 bytes less than rbp (e3d0)
Store r8d (date, 28, 1c) in the address that's 102 bytes less than rbp (e3d0)

```
0x7fffffffe370: 0x00000340      0x0000001c      0x0000000c      0x0000076f
0x7fffffffe380: 0x0040202a      0x00000000      0xffffe3e0      0x00007fff
0x7fffffffe390: 0x6e616c41      0x72755420      0x01676e69      0x0000006e
0x7fffffffe3a0: 0xffffe490      0x00007fff      0x00401365      0x00000000
0x7fffffffe3b0: 0x6e616c41      0x72755420      0x00676e69      0x00000000
0x7fffffffe3c0: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffffffe3d0: 0xffffe490      0x00007fff      0x004013ad      0x00000000
0x7fffffffe3e0: 0x6e616c41      0x72755420      0x00676e69      0x00000000
0x7fffffffe3f0: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffffffe400: 0x00000778      0x00000006      0x00000017      0x00000000
0x7fffffffe410: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffffffe420: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffffffe430: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffffffe440: 0x6e616c41      0x72755420      0x00676e69      0x00000000
0x7fffffffe450: 0x00000000      0x00000000      0x00000000      0x00000000
```

P4.b2

Wednesday, March 15, 2023 8:56 AM

401178 - 401184: move values in registers around:

Copy year to eax
Copy eax to the address 32 bytes less than rbp
Copy month to eax
Copy eax to the address 28 bytes less than rbp
Copy date to eax
Copy eax to the address 24 bytes less than rbp

This is effectively creating a copy of the birth date at a different place on the stack

4011aa - 4011df is the return process.
We move around the values in the registers and then put them In order after rax, with which we use to reference all those values.

To summarize: big parameters and return values are stored on the stack and is referenced by an address stored in a register.

P5

Friday, March 10, 2023 3:41 AM

4. I like to play D&D (Dungeons and Dragons)

The dungeon master provides a scene, and the players describe what their characters want to do, then they roll dice to determine the outcome.