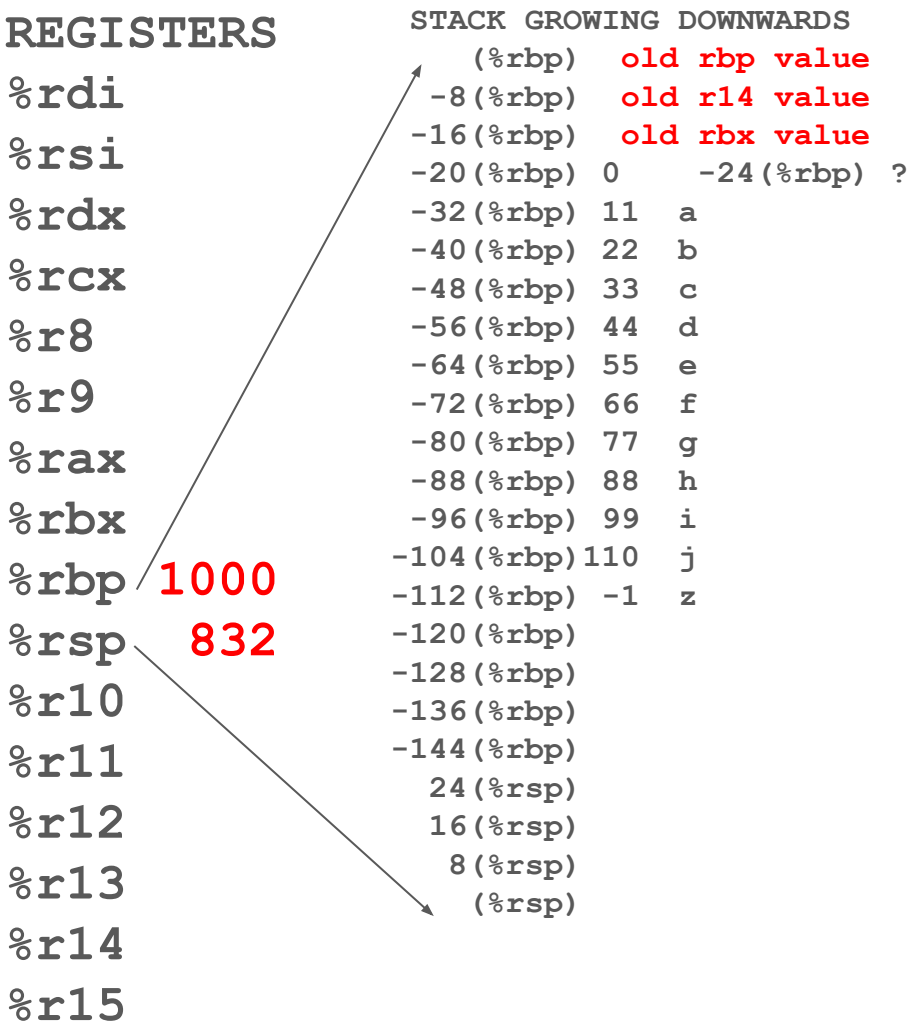
Stack Frame Tracing

3/6/2024 CS150a

```
#include <stdio.h>  // program to be compiled to X86-64
long test();

int main(void)
{ long a=11; long b=22; long c= 33; long d= 44; long e= 55; long f= 66;
  long g= 77; long h= 88; long i= 99; long j= 110;  long z = -1;
  z=test(a,b,c,d,e,f,g,h,i,j);
  printf("z=%ld\n",z);return 1;
}

long test(long a, long b, long c, long d, long e, long f, long g, long h, long i, long j){
    long x =  a+b+c+d+e+f+g+h+i+j;
    long y =  a+b+c+d+e;
    long z =  x-y;
    return z;
}
```

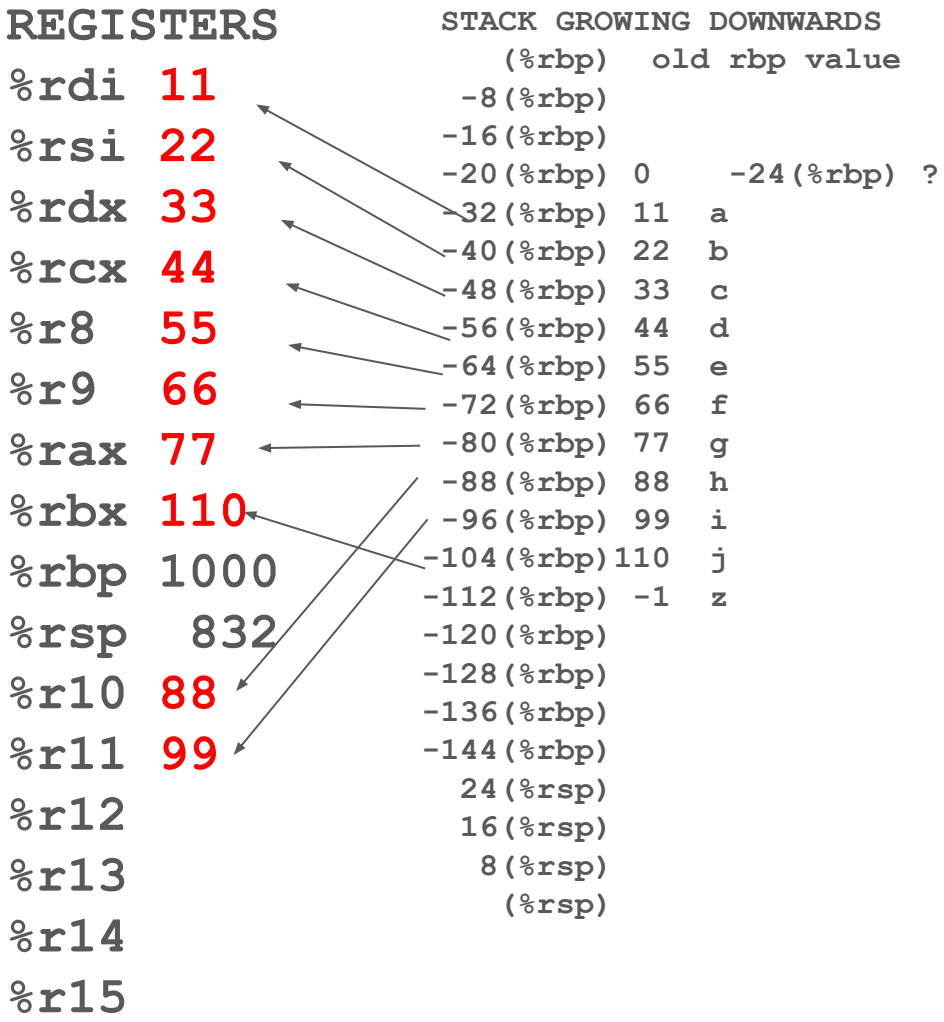


CREATE STACK FRAME - SET FRAME AND STACK POINTERS

```
int main(void)
{ long a=11; long b=22; long c= 33; long d= 44; long e= 55; long f= 66;
  long g= 77; long h= 88; long i= 99; long j= 110;  long z = -1;
  z=test(a,b,c,d,e,f,g,h,i,j);
  printf("z=%ld\n",z);return 1;
}
```

```
_main:                          ## @main
## %bb.0:          ## PROLOGUE
pushq %rbp        ## push base of previous frame into stack
movq  %rsp, %rbp   ## store rsp as base of current frame
pushq %r14           ## store callee saved registers
pushq %rbx            ## that are used below
subq   $144, %rsp ## CREATE STACK FRAME (18 longs)

        movl   $0, -20(%rbp)
        movq  $11, -32(%rbp)
        movq  $22, -40(%rbp)
        movq  $33, -48(%rbp)
        movq  $44, -56(%rbp)
        movq  $55, -64(%rbp)
        movq  $66, -72(%rbp)
        movq  $77, -80(%rbp)
        movq  $88, -88(%rbp)
        movq  $99, -96(%rbp)
        movq  $110, -104(%rbp)
        movq  $-1, -112(%rbp)
```

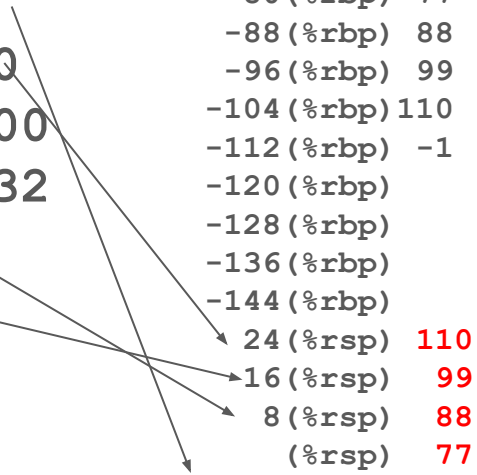


PREPARE FOR CALL TO TEST FUNCTION
MOVE LOCAL VARIABLE VALUES INTO REGISTERS

```
int main(void)
{ long a=11; long b=22; long c= 33; long d= 44; long e= 55; long f= 66;
  long g= 77; long h= 88; long i= 99; long j= 110;  long z = -1;
  z=test(a,b,c,d,e,f,g,h,i,j);
  printf("z=%ld\n",z);return 1;
}
 # move params from stack to registers
        movq  -32(%rbp), %rdi
        movq  -40(%rbp), %rsi
        movq  -48(%rbp), %rdx
        movq  -56(%rbp), %rcx
        movq  -64(%rbp), %r8
        movq  -72(%rbp), %r9
        movq  -80(%rbp), %rax
        movq  -88(%rbp), %r10
        movq  -96(%rbp), %r11
        movq  -104(%rbp), %rbx
```

```
REGISTERS
%rdi 11
%rsi 22
%rdx 33
%rcx 44
%r8  55
%r9  66
%rax 77
%rbx 110
%rbp 1000
%rsp  832
%r10 88
%r11 99
%r12
%r13
%r14
%r15
```

```
STACK GROWING DOWNWARDS
   (%rbp)   old rbp value
 -8(%rbp)
-16(%rbp)
-20(%rbp) 0     -24(%rbp) ?
-32(%rbp) 11   a
-40(%rbp) 22   b
-48(%rbp) 33   c
-56(%rbp) 44   d
-64(%rbp) 55   e
-72(%rbp) 66   f
-80(%rbp) 77   g
-88(%rbp) 88   h
-96(%rbp) 99   i
-104(%rbp)110  j
-112(%rbp) -1  z
-120(%rbp)
-128(%rbp)
-136(%rbp)
-144(%rbp)
 24(%rsp) 110
 16(%rsp)  99
  8(%rsp)  88
   (%rsp)  77
```



MOVE LAST FOUR ARGUMENTS TO THE STACK
6 IN REGISTERS, THE REST IN THE STACK

```
int main(void)
{ long a=11; long b=22; long c= 33; long d= 44; long e= 55; long f= 66;
  long g= 77; long h= 88; long i= 99; long j= 110;  long z = -1;
  z=test(a,b,c,d,e,f,g,h,i,j);
  printf("z=%ld\n",z);return 1;
}

#  set up arguments for the call
            movq      %rax, (%rsp)
            movq      %r10, 8(%rsp)
            movq      %r11, 16(%rsp)
            movq      %rbx, 24(%rsp)
            callq     _test
```

First six arguments are in registers,
Last 4 are passed on the stack

REGISTERS

```
%rdi 11
%rsi 22
%rdx 33
%rcx 44
%r8  55
%r9  66
%rax 77
%rbx 110
%rbp 824
%rsp 816
%r10 88
%r11 99
%r12
%r13
%r14
%r15
```

STACK GROWING DOWNWARDS

```
 … previous frame …
32(%rsp) 110
24(%rsp)  99
16(%rsp)  88
 8(%rsp)  77
  (%rsp)  RETURN ADDR
```

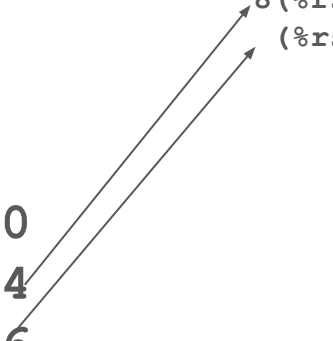
PUSH THE RETURN ADDRESS ON THE STACK
JUMP TO THE CODE FOR THE TEST FUNCTION

```
long test(long a, long b, long c, long d, long e, long f, long g, long h, long i, long j){
   long x =  a+b+c+d+e+f+g+h+i+j;
   long y =  a+b+c+d+e;
   long z =  x-y;
   return z;
}
```

PUSH the Return Address on the Stack and jump to the code for the new function!

REGISTERS

%rdi 11
%rsi 22
%rdx 33
%rcx 44
%r8 55
%r9 66
%rax 77
%rbx 110
%rbp 824
%rsp 816
%r10 88
%r11 99
%r12
%r13
%r14
%r15



STACK GROWING DOWNWARDS

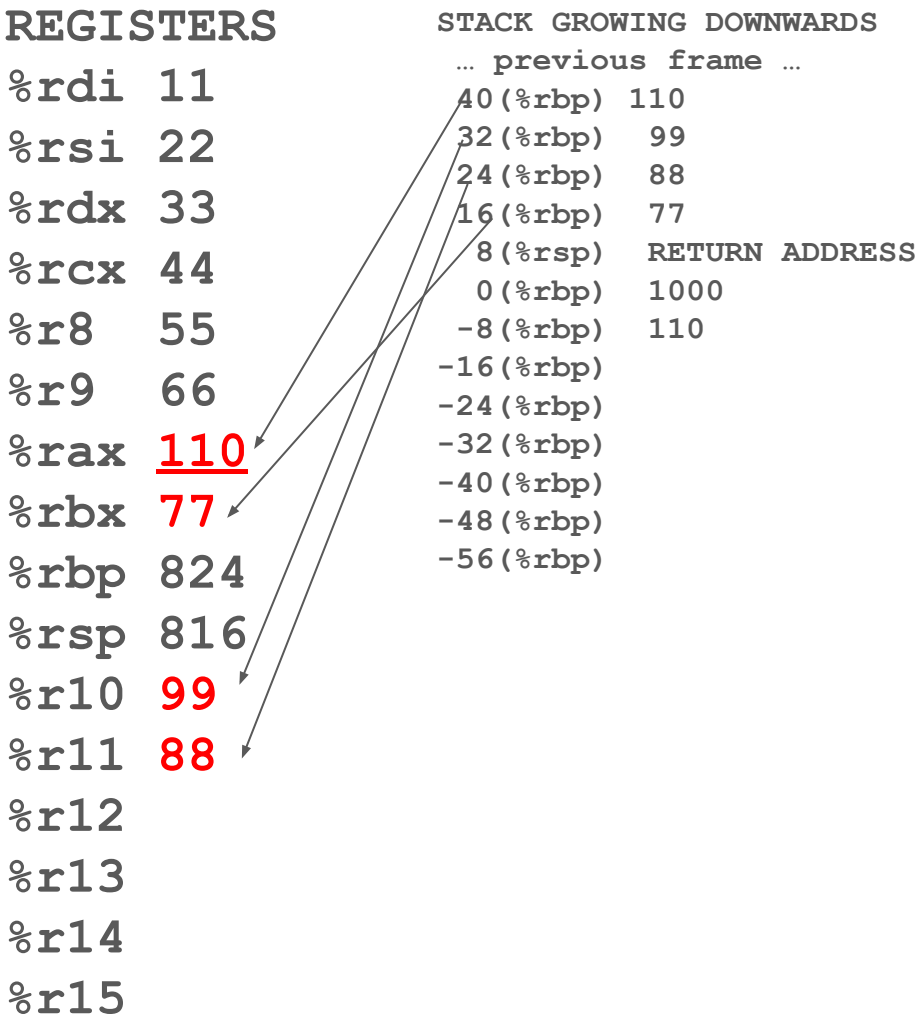
… previous frame …
48(%rbp) 110
40(%rbp) 99
32(%rbp) 88
24(%rbp) 77
16(%rsp) RETURN ADDRESS
8(%rsp) 1000
(%rsp) 110

CHANGE THE VIEW TO A NEW FRAME
STORE OLD FRAME BUFFER POINTER IN STACK
PUSH CALLEE SAVE REGISTERS YOU WILL NEED TO USE LATER

```
long test(long a, long b, long c, long d, long e, long f, long g, long h, long i, long j){
   long x =  a+b+c+d+e+f+g+h+i+j;
   long y =  a+b+c+d+e;
   long z =  x-y;
   return z;
}
```

PUSH old frame pointer onto the stack
Set new frame pointer to point to that cell
Push rbx onto the stack

```
#Prologue
_test:
pushq   %rbp
movq    %rsp, %rbp
pushq   %rbx
```



COPY PARAMETERS FROM STACK INTO REGISTERS
SO THEY CAN BE USED MORE EASILY

```
long test(long a, long b, long c, long d, long e, long f, long g, long h, long i, long j){
   long x =  a+b+c+d+e+f+g+h+i+j;
   long y =  a+b+c+d+e;
   long z =  x-y;
   return z;
}
```

We now show the stack locations with respect to %rbp
The pointer for the current frame

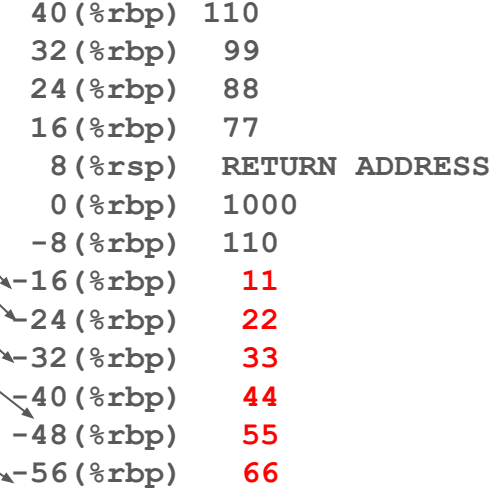
Then we copy the parameters from the stack into registers

```
movq    40(%rbp), %rax
movq    32(%rbp), %r10
movq    24(%rbp), %r11
movq    16(%rbp), %rbx

movq    %rdi, -16(%rbp)
movq    %rsi, -24(%rbp)
movq    %rdx, -32(%rbp)
movq    %rcx, -40(%rbp)
movq    %r8, -48(%rbp)
movq    %r9, -56(%rbp)
```

REGISTERS

%rdi 11
%rsi 22
%rdx 33
%rcx 44
%r8 55
%r9 66
%rax 110
%rbx 77
%rbp 824
%rsp 816
%r10 99
%r11 88
%r12
%r13
%r14
%r15



STACK GROWING DOWNWARDS

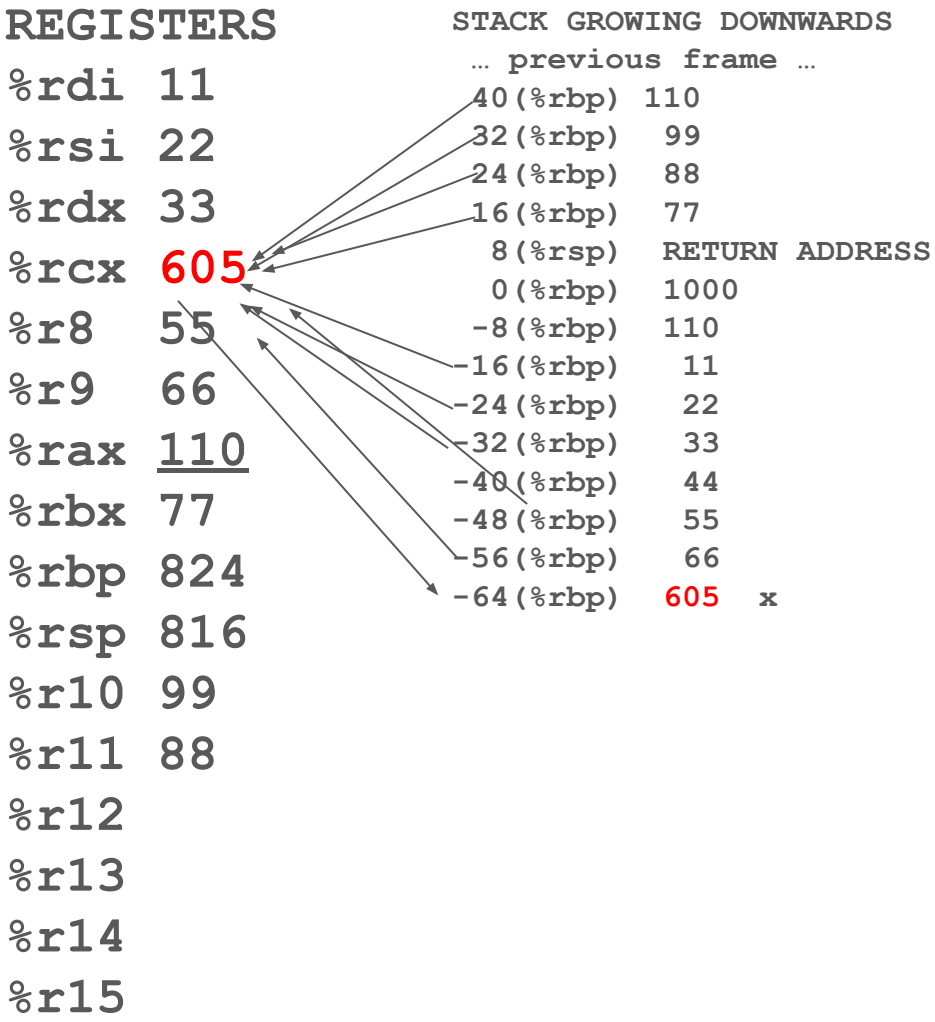
… previous frame …
40(%rbp) 110
32(%rbp) 99
24(%rbp) 88
16(%rbp) 77
8(%rsp) RETURN ADDRESS
0(%rbp) 1000
-8(%rbp) 110
-16(%rbp) 11
-24(%rbp) 22
-32(%rbp) 33
-40(%rbp) 44
-48(%rbp) 55
-56(%rbp) 66

STORE PARAMETERS PASSED IN REGISTERS TO THE STACK IN CASE YOU NEED THEM LATER AFTER A PROCEDURE CALL

```
long test(long a, long b, long c, long d, long e, long f, long g, long h, long i, long j){
   long x =  a+b+c+d+e+f+g+h+i+j;
   long y =  a+b+c+d+e;
   long z =  x-y;
   return z;
}
```

```
# Copy parameters from registers into the stack

movq    %rdi, -16(%rbp)
movq    %rsi, -24(%rbp)
movq    %rdx, -32(%rbp)
movq    %rcx, -40(%rbp)
movq    %r8, -48(%rbp)
movq    %r9, -56(%rbp)
```

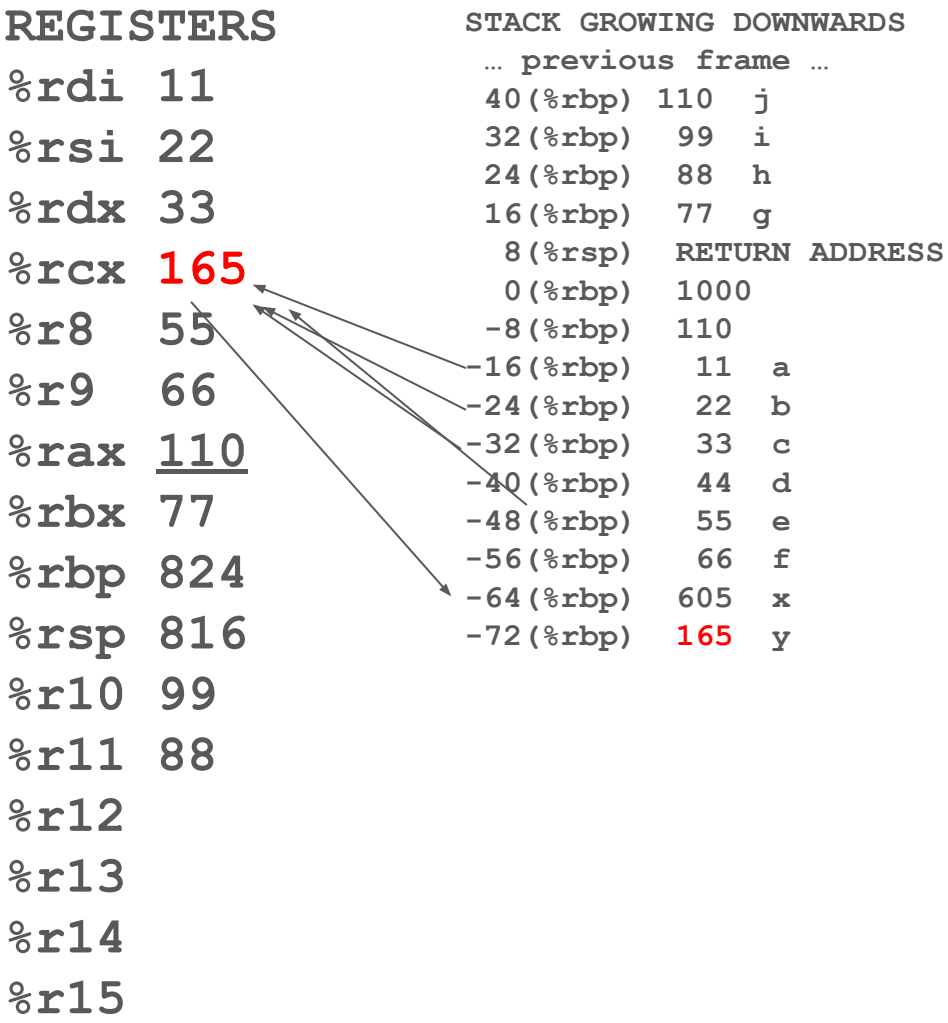


CALCULATE THE SUM USING BY COPYING FIRST VALUE TO A REGISTER THEN ADDING OTHER STACK ELEMENTS TO IT

```
long test(long a, long b, long c, long d, long e, long f, long g, long h, long i, long j){
   long x =  a+b+c+d+e+f+g+h+i+j;
   long y =  a+b+c+d+e;
   long z =  x-y;
   return z;
}
```

Next we calculate x which is the sum the values of the parameters. Ths sum is accumulated in rcx and then stored the result in the stack

```
movq   -16(%rbp), %rcx
addq   -24(%rbp), %rcx
addq   -32(%rbp), %rcx
addq   -40(%rbp), %rcx
addq   -48(%rbp), %rcx
addq   -56(%rbp), %rcx
addq   16(%rbp), %rcx
addq   24(%rbp), %rcx
addq   32(%rbp), %rcx
addq   40(%rbp), %rcx
movq   %rcx, -64(%rbp)
```



COMPUTE THE SECOND SUM, FOR LOCAL VARIABLE Y IN THE SAME WAY

```
long test(long a, long b, long c, long d, long e, long f, long g, long h, long i, long j){
   long x =  a+b+c+d+e+f+g+h+i+j;
   long y =  a+b+c+d+e;
   long z =  x-y;
   return z;
}
```

Next we calculate y = a+b+c+d+e and store it in the stack

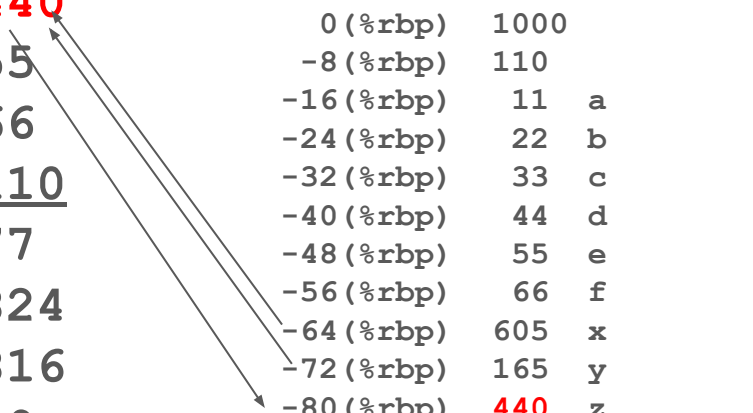
```
movq    -16(%rbp), %rcx
addq    -24(%rbp), %rcx
addq    -32(%rbp), %rcx
addq    -40(%rbp), %rcx
addq    -48(%rbp), %rcx
movq    %rcx, -72(%rbp)
```

REGISTERS

```
%rdi 11
%rsi 22
%rdx 33
%rcx 440
%r8  55
%r9  66
%rax 110
%rbx 77
%rbp 824
%rsp 816
%r10 99
%r11 88
%r12
%r13
%r14
%r15
```

STACK GROWING DOWNWARDS

```
 … previous frame …
 40(%rbp) 110  j
 32(%rbp)  99  i
 24(%rbp)  88  h
 16(%rbp)  77  g
  8(%rsp)  RETURN ADDRESS
  0(%rbp)  1000
 -8(%rbp)  110
-16(%rbp)   11  a
-24(%rbp)   22  b
-32(%rbp)   33  c
-40(%rbp)   44  d
-48(%rbp)   55  e
-56(%rbp)   66  f
-64(%rbp)  605  x
-72(%rbp)  165  y
-80(%rbp)  440  z
```



COMPUTE THE DIFFERENCE OF X AND Y AND STORE IN Z

```
long test(long a, long b, long c, long d, long e, long f, long g, long h, long i, long j){
   long x =  a+b+c+d+e+f+g+h+i+j;
   long y =  a+b+c+d+e;
   long z =  x-y;
   return z;
}
```

Then we calculate z = x-y and store it in the stack

```
movq    -64(%rbp), %rcx
subq    -72(%rbp), %rcx
movq    %rcx, -80(%rbp)
```

And we move it back to the register (not optimial1)

```
movq    -80(%rbp), %rcx
```

REGISTERS

```
%rdi 11
%rsi 22
%rdx 33
%rcx 440
%r8  55
%r9  66
%rax 110440
%rbx 77
%rbp 824
%rsp 816
%r10 99
%r11 88
%r12
%r13
%r14
%r15
```

STACK GROWING DOWNWARDS

```
 … previous frame …
 40(%rbp) 110  j
 32(%rbp)  99  i
 24(%rbp)  88  h
 16(%rbp)  77  g
  8(%rbp)  RETURN ADDRESS
  0(%rbp)  1000
 -8(%rbp)  110
-16(%rbp)   11  a
-24(%rbp)   22  b
-32(%rbp)   33  c
-40(%rbp)   44  d
-48(%rbp)   55  e
-56(%rbp)   66  f
-64(%rbp)  605  x
-72(%rbp)  165  y
-80(%rbp)  440  z
-88(%rbp)  110
```

MOVE Z VALUE INTO THE RAX REGISTER
WHICH IS THE REGISTER FOR THE RETURN VALUE

```
long test(long a, long b, long c, long d, long e, long f, long g, long h, long i, long j){
   long x =  a+b+c+d+e+f+g+h+i+j;
   long y =  a+b+c+d+e;
   long z =  x-y;
   return z;
}
```

Then we store the return value in rax
(but first move the current value into the stack, in case we need it later)

```
movq    %rax, -88(%rbp)      ## 8-byte Spill
movq    %rcx, %rax
```

```
REGISTERS
%rdi 11
%rsi 22
%rdx 33
%rcx 440
%r8  55
%r9  66
%rax 440
%rbx 110
%rbp 1000
%rsp 816
%r10 99
%r11 88
%r12
%r13
%r14
%r15
```

```
STACK GROWING DOWNWARDS
 … previous frame …
32(%rsp) 110  j
24(%rsp)  99  i
16(%rsp)  88  h
 8(%rsp)  77  g
 0(%rsp)  RETURN ADDRESS
```

RESTORE THE CALLEE SAVE REGISTERS
AND JUMP BACK TO THE PARENT FUNCTION

```
long test(long a, long b, long c, long d, long e, long f, long g, long h, long i, long j){
   long x =  a+b+c+d+e+f+g+h+i+j;
   long y =  a+b+c+d+e;
   long z =  x-y;
   return z;
}
```

And we reset the value of rbx (a callee save register)

```
popq    %rbx
```

And we reset the frame pointer to point to the beginning of the previous frqame

```
popq    %rbp
```

We have now also reset the stack pointer %rsp
So we can use the RETURN ADDRESS at (%rsp)
To get back to the previous function

```
retq
```

```
REGISTERS
%rdi 11
%rsi 440
%rdx 33
%rcx 44
%r8  55
%r9  66
%rax 440
%rbx 110
%rbp 1000
%rsp  816
%r10 99
%r11 88
%r12
%r13
%r14
%r15
```

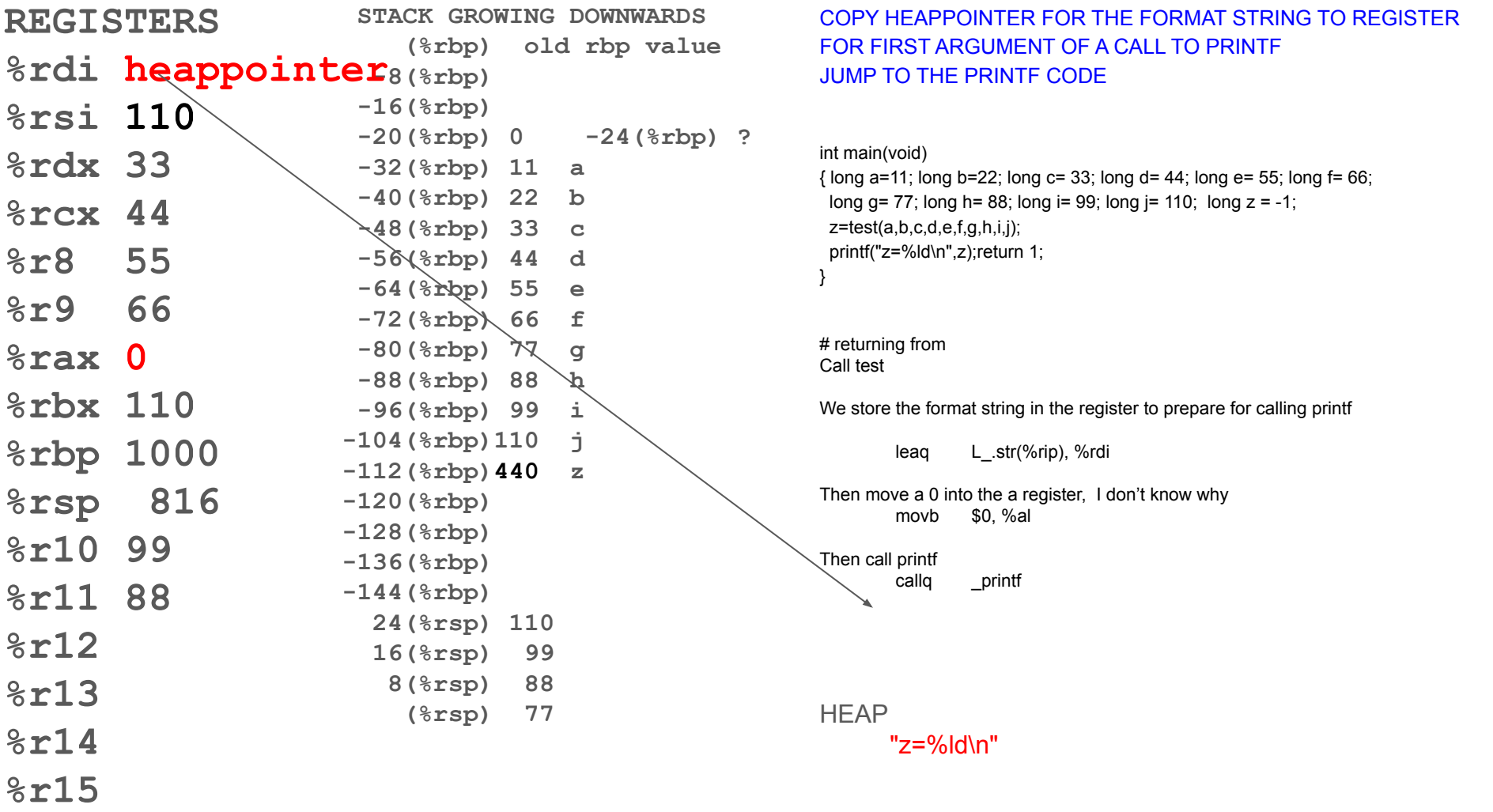
```
STACK GROWING DOWNWARDS
   (%rbp)   old rbp value
 -8(%rbp)
-16(%rbp)
-20(%rbp) 0     -24(%rbp) ?
-32(%rbp) 11   a
-40(%rbp) 22   b
-48(%rbp) 33   c
-56(%rbp) 44   d
-64(%rbp) 55   e
-72(%rbp) 66   f
-80(%rbp) 77   g
-88(%rbp) 88   h
-96(%rbp) 99   i
-104(%rbp)110  j
-112(%rbp)440  z
-120(%rbp)
-128(%rbp)
-136(%rbp)
-144(%rbp)
  24(%rsp) 110
  16(%rsp)  99
   8(%rsp)  88
    (%rsp)  77
```

STORE RETURN VALUE (IN RAX) TO THE STACK
MOVE IT TO POSITION 2 OF THE CALL TO PRINTF (I.E. RSI)

```
int main(void)
{ long a=11; long b=22; long c= 33; long d= 44; long e= 55; long f= 66;
  long g= 77; long h= 88; long i= 99; long j= 110;  long z = -1;
  z=test(a,b,c,d,e,f,g,h,i,j);
  printf("z=%ld\n",z);return 1;
}
```

```
callq       _test  # We have just returned from this call

# store the return value in the stack
            movq      %rax, -112(%rbp)
# and copy it to rsi to set up for the PRINT call
            movq      -112(%rbp), %rsi
```


REGISTERS
%rdi heappointer
%rsi 110
%rdx 33
%rcx 44
%r8 55
%r9 66
%rax 0
%rbx 110
%rbp 1000
%rsp 816
%r10 99
%r11 88
%r12
%r13
%r14
%r15
STACK GROWING DOWNWARDS
(%rbp) old rbp value
8(%rbp)
-16(%rbp)
-20(%rbp) 0 -24(%rbp) ?
-32(%rbp) 11 a
-40(%rbp) 22 b
-48(%rbp) 33 c
-56(%rbp) 44 d
-64(%rbp) 55 e
-72(%rbp) 66 f
-80(%rbp) 77 g
-88(%rbp) 88 h
-96(%rbp) 99 i
-104(%rbp) 110 j
-112(%rbp) 440 z
-120(%rbp)
-128(%rbp)
-136(%rbp)
-144(%rbp)
24(%rsp) 110
16(%rsp) 99
8(%rsp) 88
(%rsp) 77
COPY HEAPPOINTER FOR THE FORMAT STRING TO REGISTER
FOR FIRST ARGUMENT OF A CALL TO PRINTF
JUMP TO THE PRINTF CODE
int main(void)
{ long a=11; long b=22; long c= 33; long d= 44; long e= 55; long f= 66;
long g= 77; long h= 88; long i= 99; long j= 110; long z = -1;
z=test(a,b,c,d,e,f,g,h,i,j);
printf("z=%ld\n",z);return 1;
}
# returning from
Call test
We store the format string in the register to prepare for calling printf
leaq L_.str(%rip), %rdi
Then move a 0 into the a register, I don't know why
movb $0, %al
Then call printf
callq _printf
HEAP
"z=%ld\n"

```
REGISTERS
%rdi heappointer
%rsi 110
%rdx 33
%rcx 44
%r8  55
%r9  66
%rax  1
%rbx 110
%rbp 1000
%rsp  816
%r10 99
%r11 88
%r12
%r13
%r14  1
%r15
```

```
STACK GROWING DOWNWARDS
   (%rbp)   old rbp value
  8(%rbp)
-16(%rbp)
-20(%rbp)  0      -24(%rbp)  ?
-32(%rbp)  11   a
-40(%rbp)  22   b
-48(%rbp)  33   c
-56(%rbp)  44   d
-64(%rbp)  55   e
-72(%rbp)  66   f
-80(%rbp)  77   g
-88(%rbp)  88   h
-96(%rbp)  99   i
-104(%rbp)110   j
-112(%rbp)110   z
-116(%rbp)  0   -120(%rbp)  0
-128(%rbp)
-136(%rbp)
-144(%rbp)
  24(%rsp) 110
  16(%rsp)  99
   8(%rsp)  88
    (%rsp)  77
```

Move 1 into the return value position EAX
(but first save old value of RAX on the stack)

```
int main(void)
{ long a=11; long b=22; long c= 33; long d= 44; long e= 55; long f= 66;
  long g= 77; long h= 88; long i= 99; long j= 110;  long z = -1;
  z=test(a,b,c,d,e,f,g,h,i,j);
  printf("z=%ld\n",z);return 1;
}
```

We've just returned from the call to printf

EPILOG, we need to return 1, so
We have to move it into rax, but rax is a callee save
So we need to move rax into the stack first

```
movl    $1, %r14d   # move 1 to r14d
movl    %eax, -116(%rbp)
movl    %r14d, %eax
```

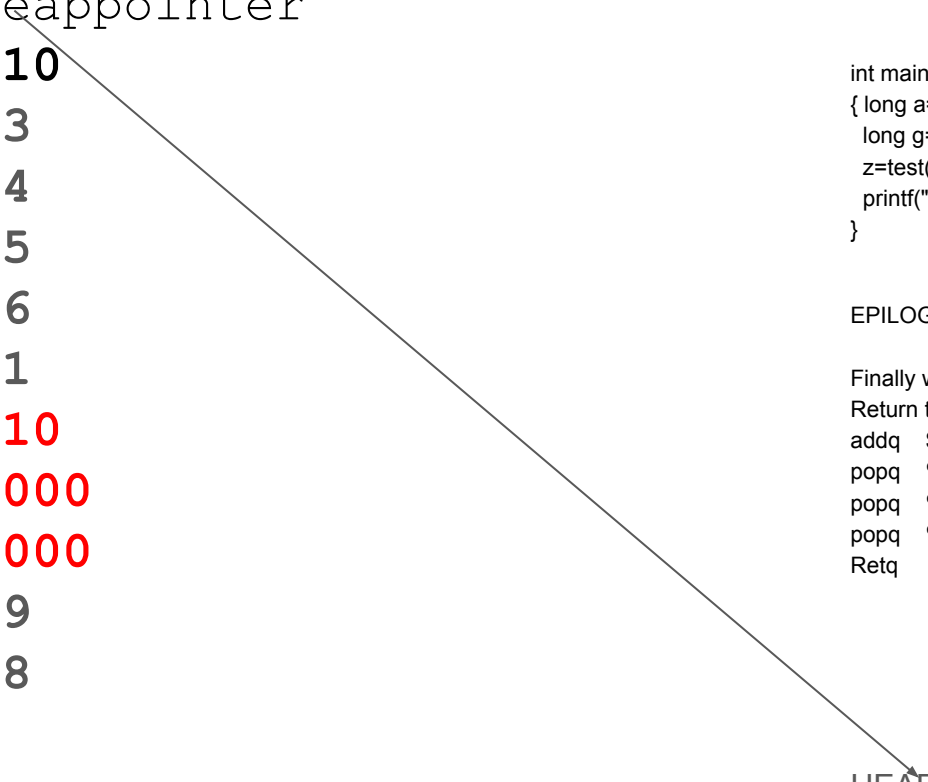
HEAP

"z=%ld\n"

REGISTERS

%rdi heappointer
%rsi 110
%rdx 33
%rcx 44
%r8 55
%r9 66
%rax 1
%rbx 110
%rbp 1000
%rsp 1000
%r10 99
%r11 88
%r12
%r13
%r14
%r15

STACK GROWING DOWNWARDS
(%rsp)



HEAP

"z=%ld\n"

RESET THE STACK AND FRAME POINTERS AND
RETURN CONTROL TO THE OPERATING SYSTEM

```
int main(void)
{ long a=11; long b=22; long c= 33; long d= 44; long e= 55; long f= 66;
  long g= 77; long h= 88; long i= 99; long j= 110;  long z = -1;
  z=test(a,b,c,d,e,f,g,h,i,j);
  printf("z=%ld\n",z);return 1;
}
```

EPILOG, we need to return 1, so

Finally we reset the stack pointer,
Return the values of rbx, r14, and rbp and return to the Operating System

```
addq    $144, %rsp
popq    %rbx
popq    %r14
popq    %rbp
Retq
```

REGISTERS

%rdi
%rsi
%rdx
%rcx
%r8
%r9
%rax
%rbx
%rbp
%rsp
%r10
%r11
%r12
%r13
%r14
%r15

STACK GROWING DOWNWARDS

(%rbp)
-8(%rbp)
-16(%rbp)
-24(%rbp)
-32(%rbp)
-40(%rbp)
-48(%rbp)
-56(%rbp)
-64(%rbp)
-72(%rbp)
-80(%rbp)
-88(%rbp)
-96(%rbp)
-104(%rbp)
-112(%rbp)
-120(%rbp)
-128(%rbp)
-136(%rbp)
-144(%rbp)
24(%rsp)
16(%rsp)
8(%rsp)
(%rsp)

CODE:

REGISTERS

%rdi
%rsi
%rdx
%rcx
%r8
%r9
%rax
%rbx
%rbp
%rsp
%r10
%r11
%r12
%r13
%r14
%r15

STACK GROWING DOWNWARDS

(%rbp)
-8(%rbp)
-16(%rbp)
-24(%rbp)
-32(%rbp)
-40(%rbp)
-48(%rbp)
-56(%rbp)
-64(%rbp)
-72(%rbp)
-80(%rbp)
-88(%rbp)
-96(%rbp)
-104(%rbp)
-112(%rbp)
-120(%rbp)
-128(%rbp)
-136(%rbp)
-144(%rbp)
24(%rsp)
16(%rsp)
8(%rsp)
(%rsp)

CODE:

REGISTERS

%rdi
%rsi
%rdx
%rcx
%r8
%r9
%rax
%rbx
%rbp
%rsp
%r10
%r11
%r12
%r13
%r14
%r15

STACK GROWING DOWNWARDS

(%rbp)
-8(%rbp)
-16(%rbp)
-24(%rbp)
-32(%rbp)
-40(%rbp)
-48(%rbp)
-56(%rbp)
-64(%rbp)
-72(%rbp)
-80(%rbp)
-88(%rbp)
-96(%rbp)
-104(%rbp)
-112(%rbp)
-120(%rbp)
-128(%rbp)
-136(%rbp)
-144(%rbp)
24(%rsp)
16(%rsp)
8(%rsp)
(%rsp)

CODE:

REGISTERS

%rdi
%rsi
%rdx
%rcx
%r8
%r9
%rax
%rbx
%rbp
%rsp
%r10
%r11
%r12
%r13
%r14
%r15

STACK GROWING DOWNWARDS

(%rbp)
-8(%rbp)
-16(%rbp)
-24(%rbp)
-32(%rbp)
-40(%rbp)
-48(%rbp)
-56(%rbp)
-64(%rbp)
-72(%rbp)
-80(%rbp)
-88(%rbp)
-96(%rbp)
-104(%rbp)
-112(%rbp)
-120(%rbp)
-128(%rbp)
-136(%rbp)
-144(%rbp)
24(%rsp)
16(%rsp)
8(%rsp)
(%rsp)

CODE: