1. Table

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Line number** | **PC** | **Instruction** | **%rdi** | **%rsi** | **%rax** | **%rsp** | **\*%rsp** | **Description** |
| 14 | 4004fb | callq | 3 | 9 | 0 | 0x7fffffffe820 | --- | Call top(x,y) |
| --- |  |  |  |  |  |  |  |  |
| 6 | 4004dd | add | 3 | 9 | 0 | 0x7fffffffe818 | 400500 | Entry of top |
| 7 | 4004e0 | mov | 12 | 9 | 0 | 0x7fffffffe818 | 400500 | Move 0 to lower half of rax |
| 8 | 4004e5 | callq | 12 | 9 | 0 | 0x7fffffffe818 | 400500 | Call leaf(x,y) |
| --- |  |  |  |  |  |  |  |  |
| 2 | 4004d6 | mov | 12 | 9 | 0 | 0x7fffffffe810 | 4004ea | Entry of leaf |
| 3 | 4004d9 | sub | 12 | 9 | 12 | 0x7fffffffe810 | 4004ea | Subtract y from z |
| 4 | 4004dc | retq | 12 | 9 | 3 | 0x7fffffffe810 | 4004ea | Return 3 from leaf |
| --- |  |  |  |  |  |  |  |  |
| 9 | 4004ea | repz retq | 12 | 9 | 3 | 0x7fffffffe818 | 400500 | Return 3 from top |
| --- |  |  |  |  |  |  |  |  |
| 15 | 400500 | repz retq | 12 | 9 | 3 | 0x7fffffffe820 | --- | Return 3 from main |

1. Q&A
2. 32 Bytes
3. Line 24 (sub $0x20, %rsp) allocates and line 8b (add $0x20, %rsp) frees the local stack
4. %rbx – d (mov %rax,%rbx), %rbp – e (mov %rax,%rbp)
5. rdi - mov %rdi,0x18(%rsp), rsi - mov %rsi,0x10(%rsp), rdx - mov %rdx,0x8(%rsp)
6. All of the local variables cannot be stored in callee-saved registers because there are a limited number of these and some of these are reserved for function return and arguments. Therefore, some of the local variables cannot be stored in the callee-stored variables.
7. Fill in the blank + Q&A
   1. The unknown() function stores

long unknown(unsigned long x) {

if (**x == 0**)

return **0;**

unsigned long nx = **(x ^ 1) & 1;**

long rv = unknown(x **>> 1);**

return **rv + nx;**

}

1. Table

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Array declaration** | | **Element size** | **Total size** | **Start address** | | | **Element *i*** |
| (a) | char r[4]; | 1 byte | | 4 bytes | | x\_r | X\_r + i | |
| (b) | char \*s[4]; | 8 bytes | | 32 bytes | | x\_s | X\_s + 8i | |
| (c) | short t[5]; | 2 bytes | | 10 bytes | | x\_t | X\_t + 2i | |
| (d) | short \*u[5]; | 8 bytes | | 40 bytes | | x\_u | X\_u + 8i | |
| (e) | short \*\*v[3]; | 8 bytes | | 24 bytes | | x\_v | X\_v + 8i | |
| (f) | int w[4]; | 4 bytes | | 16 bytes | | x\_w | X\_w + 4i | |
| (g) | long \*x[5]; | 8 bytes | | 40 bytes | | x\_x | X\_x + 8i | |
| (h) | double \*y[6]; | 8 bytes | | 48 bytes | | x\_y | X\_y + 8i | |

1. Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Expression** | **Type** | **Value** | **Assembly code** |
| (a) | S[2] | Short | M[x\_s + 4] | movw 4(%rdx), %ax |
| (b) | S+2 | Short \* | x\_s + 4 | leaq 4(%rdx), %rax |
| (c) | &S[i] | Short \* | x\_s + 2i | leaq (%rdx, %rcx, 2), %rax |
| (d) | S[2\*i+1] | Short | M[x\_s + 4i + 2] | movw 2(%rdx, %rcx, 4), %ax |
| (e) | S+i-2 | Short \* | x\_s + 2i – 4 | leaq -4(%rdx, %rcx, 2), %rax |
| (f) | \*(S+i-2) | Short | M[x\_s + 2i – 4] | movw -4(%rdx, %rcx, 2), %ax |
| (g) | S+(++i)+2 | Short \* | x\_s + 2\*(i+1) + 4  x\_s + 2i + 6 | leaq 6(%rdx, %rcx, 2), %rax |
| (h) | \*(S+(i++)+2) | Short | M[x\_s + 2i + 6) | movw 6(%rdx, %rcx, 2), %ax |
| (i) | \*S-- | Short | M[x\_s] | movw %rdx, %ax |
| (j) | \*(S--) | Short | M[x\_s] | movw %rdx, %ax |