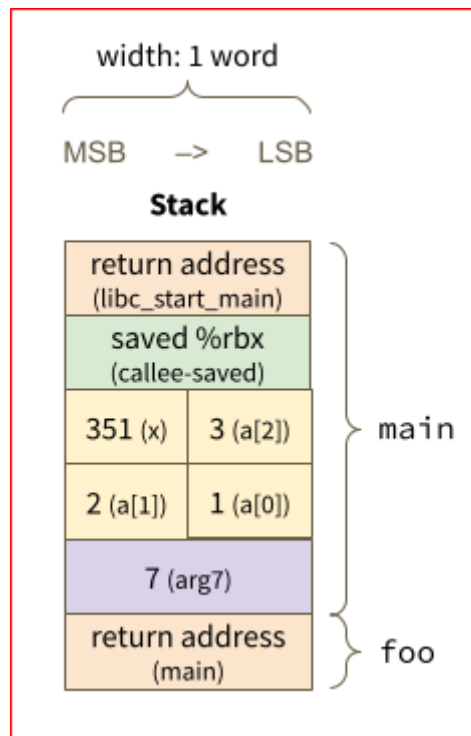


CSE 351 Section 5 – Calling Convention & Stack Discipline

Stack Frame Example

Consider the following lines of code and draw out the stack frames for `main` and `foo` right before `foo` returns (i.e., before any deallocation):

```
int main(int argc, char* argv[]) {  
    int x = 351;  
    int a[] = {1, 2, 3};  
    int y = foo(&x, 2, 3, 4, 5, 6, 7);  
    return y + argc;  
}  
  
int foo(int* arg1, int arg2, ..., int arg7) {  
    return *arg1 + arg7;  
}
```



Stack Exercise

Consider the following x86-64 assembly and C code for the recursive function r fun.

```
// Recursive function rfun
long rfun(char* s) {
    if (*s) {
        long temp = (long)*s;
        s++;
        return temp + rfun(s);
    }
    return 0;
}

// Main Function - program entry
int main(int argc, char** argv) {
    char* s = "351";
    long r = rfun(s);
    printf("r: %ld\n", r);
}
```

```
0000000000401126 <rfun>:
401126: 0f b6 07          movzbl (%rdi),%eax
401129: 84 c0             test    %al,%al
40112b: 75 06             jne     401133 <rfun+0xd>
40112d: b8 00 00 00 00    mov     $0x0,%eax
401132: c3               retq
401133: 53               push    %rbx
401134: 48 0f be d8       movsbq %al,%rbx
401138: 48 83 c7 01       add     $0x1,%rdi
40113c: e8 e5 ff ff ff    callq   401126 <rfun>
401141: 48 01 d8          add     %rbx,%rax
401144: 5b               pop     %rbx
401145: c3               retq
```

a) In terms of the C function, what value is being saved on the stack?

temp, stored in %rbx. The old value needs to be saved before the register is overwritten.

b) What is the return address to r fun that gets stored on the stack during the recursive calls (in hex)?

0x401141

- c) Assume `main` calls `r fun` with `char* s = "351"` and then prints the result using the `printf` function, as shown in the C code above. Assume `printf` does not call any other procedure. Starting with (and including) `main`, answer the following.

`main` $+(r\text{ fun} * 4) + \text{printf}$

Total stack frames created: **6**

Maximum stack depth (in frames): **5**

- d) Assume `main` calls `r fun` with `char* s = "351"`, as shown in the C code. After `main` calls `r fun`, we find that the return address to `main` is stored on the stack at address `0x7fffffffde88`. On the first call to `r fun`, the register `%rdi` holds the address `0x402010`, which is the address of the input string `"351"` (i.e., `char* s = 0x402010`). For each address in the stack diagram below, fill in both the **value** and a **description** of the entry.

Memory Address	Value	Description
<code>0x7fffffffde88</code>	<code>0x401154</code>	Return address to <code>main</code>
<code>0x7fffffffde80</code>	Unknown	Original <code>%rbx</code>
<code>0x7fffffffde78</code>	<code>0x401141</code>	Return address to <code>r fun</code>
<code>0x7fffffffde70</code>	<code>'3' = 0x33</code>	Saved <code>%rbx</code>
<code>0x7fffffffde68</code>	<code>0x401141</code>	Return address to <code>r fun</code>
<code>0x7fffffffde60</code>	<code>'5' = 0x35</code>	Saved <code>%rbx</code>
<code>0x7fffffffde58</code>	<code>0x401141</code>	Return address to <code>r fun</code>