CSci 365: Organizations of Programming Languages

Name: Kenneth Jahnke

**Assignment 8 (chap. 10): 40 points**

For the drawings of the stack with ARIs, see the example on page 3 - 4.

You must clearly give the address to which each static/dynamic link point.

The starting address of a stack is 100 whose next address is increased by 1, i.e. 100, 101, 102, …

Q1. [10] Show (i.e. Draw) the stack contents with all activation record instances, including static and dynamic chains as well as others, when execution reaches position 1 in the following skeletal program. Assume bigsub is at level 1.

**function** bigsub() {

**function** a() {

**function** b() {

... <----------------------------1

} // end of b

**function** c() {

...

b();

...

} // end of c

...

c();

...

} // end of a

...

a();

...

} // end of bigsub

|  |  |  |  |
| --- | --- | --- | --- |
| Address | ARI for subprogram | field | Pointer:  Link address |
| 109 | b | Dynamic link to: c | 104 |
| 108 |  | Static link to: a | 101 |
| 107 |  | Return address to: c | 104 |
| 106 | c | Dynamic link to: a | 101 |
| 105 |  | Static link to: a | 101 |
| 104 |  | Return address to: a | 101 |
| 103 | a | Dynamic link to: bigsub | 100 |
| 102 |  | Static link to: bigsub | 100 |
| 101 |  | Return address to: bigsub | 100 |
| **100** | bigsub | Null (no locals) |  |

Q2. [10] Show (i.e. Draw) the stack with all activation record instances, including static and dynamic chains, when execution reaches position 1 in the following skeletal program. Assume bigsub is at level 1.

function bigsub() {

function a(flag) {

function b() {

...

a(false);

...

} // end of b

...

if (flag)

b();

else c();

...

} // end of a

function c() {

function d() {

... <------------------------1

} // end of d

...

d();

...

} // end of c

...

a(true);

...

} // end of bigsub

The calling sequence for this program for execution to reach d is:

bigsub calls a; a calls b; b calls a; a calls c; c calls d;

|  |  |  |  |
| --- | --- | --- | --- |
| Address | ARI for subprogram | field | Pointer:  Link address |
| 117 | d | Dynamic link to: c | 112 |
| 116 |  | Static link to: c | 112 |
| 115 |  | Return address to: c | 112 |
| 114 | c | Dynamic link to: a | 108 |
| 113 |  | Static link to: bigsub | 100 |
| 112 |  | Return address to: a | 108 |
| 111 | a | Parameter:  flag(false) |  |
| 110 |  | Dynamic link to: b | 105 |
| 109 |  | Static link to: bigsub | 100 |
| 108 |  | Return address to: b | 105 |
| 107 | b | Dynamic link to: a | 101 |
| 106 |  | Static link to: a | 101 |
| 105 |  | Return address to: a | 101 |
| 104 | a | Parameter:  flag(true) |  |
| 103 |  | Dynamic link to: | 100 |
| 102 |  | Static link to: bigsub | 100 |
| 101 |  | Return address to: | 100 |
| **100** | bigsub | Null (no locals) |  |

Q3. [10] Show the stack with all activation record instances, including static and dynamic chains, when execution reaches position 1 in the following skeletal program. This program uses the *deep-access method* to implement dynamic scoping.

void fun1() {

float a;

...

}

void fun2() {

int b, c;

... <------- 1

}

void fun3() {

float d;

... **<------- 1**

}

void main() {

char e, f, g;

...

}

The calling sequence for this program for execution to reach fun3 is

main calls fun2; fun2 calls fun1; fun1 calls fun1; fun1 calls fun3.

|  |  |  |  |
| --- | --- | --- | --- |
| Address | ARI for subprogram | field | Pointer:  Link address |
| 119 | fun3 | Local: d |  |
| 118 |  | Dynamic link to: fun1 | 112 |
| 117 |  | Static link to: global space | global |
| 116 |  | Return address to: fun1 | 112 |
| 115 | fun1 | Local: a |  |
| 114 |  | Dynamic link to: fun1 | 108 |
| 113 |  | Static link to: global space | global |
| 112 |  | Return address to: fun1 | 108 |
| 111 | fun1 | Local: a |  |
| 110 |  | Dynamic link to: fun2 | 103 |
| 109 |  | Static link to: global space | global |
| 108 |  | Return address to: fun2 | 103 |
| 107 | fun2 | Local: c |  |
| 106 |  | Local: b |  |
| 105 |  | Dynamic link to: main | 100 |
| 104 |  | Static link to: global space | global |
| 103 |  | Return address to: main | 100 |
| 102 | main | Local: g |  |
| 101 |  | Local: f |  |
| **100** |  | Local: e |  |

fun1, fun2, fun3 are defined at the same level as main – the global space.

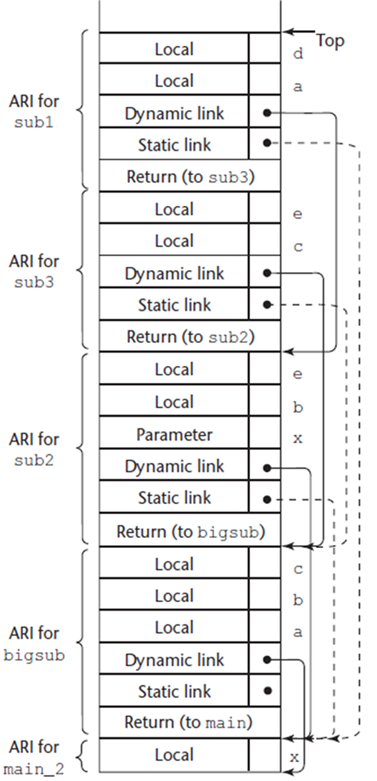
Q4. [10] Assume that the program of Q3 is implemented using the *shallow-access method* using a stack for each variable name. Show the stacks for the time of the execution of fun3, assuming execution found its way to that point through the sequence of calls shown in Q3.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| fun1 |  |  |  |  |  |  |
| fun1 | fun2 | fun2 | fun3 | main | main | main |
| a | b | c | d | e | f | g |

Example: top = 123

For a stack with the ARIs below, put it in the table with the link address instead of the arrows of the links.

Use the given table in the next page. The base address is 100.



|  |  |  |  |
| --- | --- | --- | --- |
| Address | ARI for subprogram | field | Pointer:  Link address |
| … |  | ….. | … |
| 122 |  | Local: d |  |
| 121 |  | Local: a |  |
| 120 | sub1 | Dynamic link to: sub3 | 113 |
| 119 |  | Static link to: bigsub | 101 |
| 118 |  | Return to: sub3 |  |
| 117  116 |  | Local: e  Local: c |  |
| 115 |  | Dynamic link to: sub2 | 107 |
| 114 | sub3 | Static link to: sub2 | 107 |
| 113 |  | Return address to: sub2 |  |
| 112  111 |  | Local: e  Local: b |  |
| 110 |  | Parameter: x  (value?) |  |
| 109 | sub2 | Dynamic link to: bigsub | 101 |
| 108 |  | static link to:  bigsub | 101 |
| 107 |  | Return address to:  bigsub |  |
| 106  105 |  | Local: c  Local: b |  |
| 104 |  | Local: a |  |
| 103 | bigsub | Dynamic link to: main | 100 |
| 102 |  | Static link to: main | 100 |
| 101 |  | Return address to: main | 100 |
| **100** | main | Local: x |  |