Universitetet i Oslo
Institutt for Informatikk
PMA, PSE
Axelsen, Krogdahl, Møller-Pedersen, Steffen

INF 5110: Compiler construction

Spring 2017 **Series 7** 26. 4. 2017

Topic: Run-time environments (Chapter 7)

Issued: 26. 4. 2017

Exercise 1 (Run-time environment) [1] Draw a possible organization for the runtime environment of the following C program, similar to that of Figure 7.4 on page 354 (for the gcd-example) from [1]:

1. after entry into block `A` in function `f`.
2. after entry into block `B` in function `g`.

```
1  int a[10];
2  char * s = "hello";
3
4  int f(int i, int b[])
5  { int j=i;
6   A:{ int i=j;
7       char c = b[i];
8       //...;
9     }
10    return 0;
11 }
12
13 void g(char * s)
14 { char c = s[0];
15  B:{ int a[5];
16      // ...;
17    }
18 }
19
20 main ()
21 { int x=1;
22   x = f(x,a);
23   g(s);
24   return 0;
25 }
```

Exercise 2 (Activation records) [2] Draw the stack of activation reords for the following *Pascal* program, showing the *control* and *access* links, after the second call to procedure `c`. Describe how the variable `x` is accessed from within `c`.

[1] The task corresponds to [1, Exercise 7.2.]
[2] The task corresponds to [1, Exercise 7.4.]

```
1  program env;
2
3  procedure a;
4  var x: integer;
5
6      procedure b;
7          procedure c;
8          begin
9              x := 2;
10             b;
11         end;
12     begin (* b *)
13         c;
14     end;
15
16 begin (* a *)
17     b;
18 end;
19
20 begin (* main *)
21     a;
22 end.
```

Exercise 3 (Access chaining vs. display) [3] An alternative to access chaining in a language with local procedures is to keep the access links in an array *outside* the stack, *indexed* by the *nesting level*. This array is called the *display*. For example, the run-time stacks of Figure 7.12 resp. of Figure 7.13 from the book would look as Figure 1 resp. Figure 2.

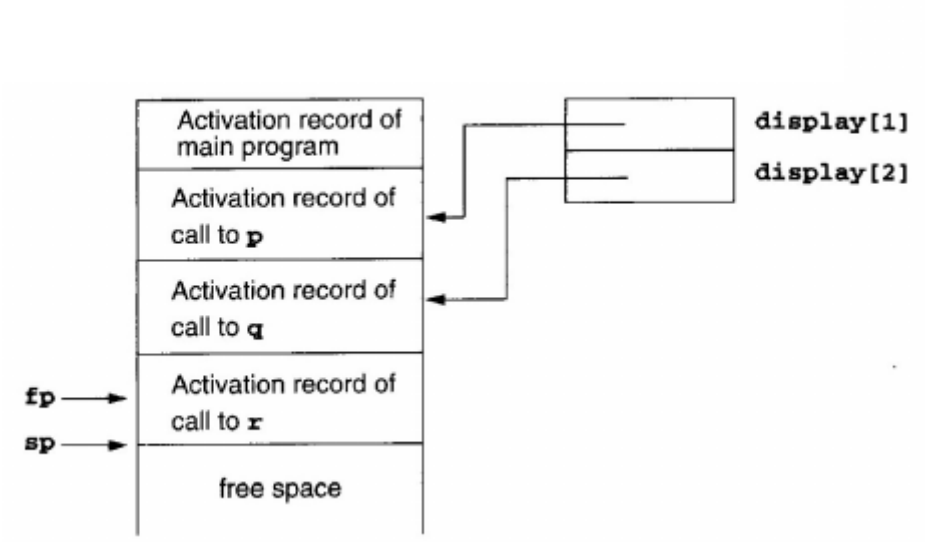

Figure 1: RTE with display (1)

1. Describe how a display can improve *efficiency* of nonlocal references from deeply nested procedures
2. Redo Exercise 2 using a display.

[3] The task corresponds to [1, Exercise 7.10.]

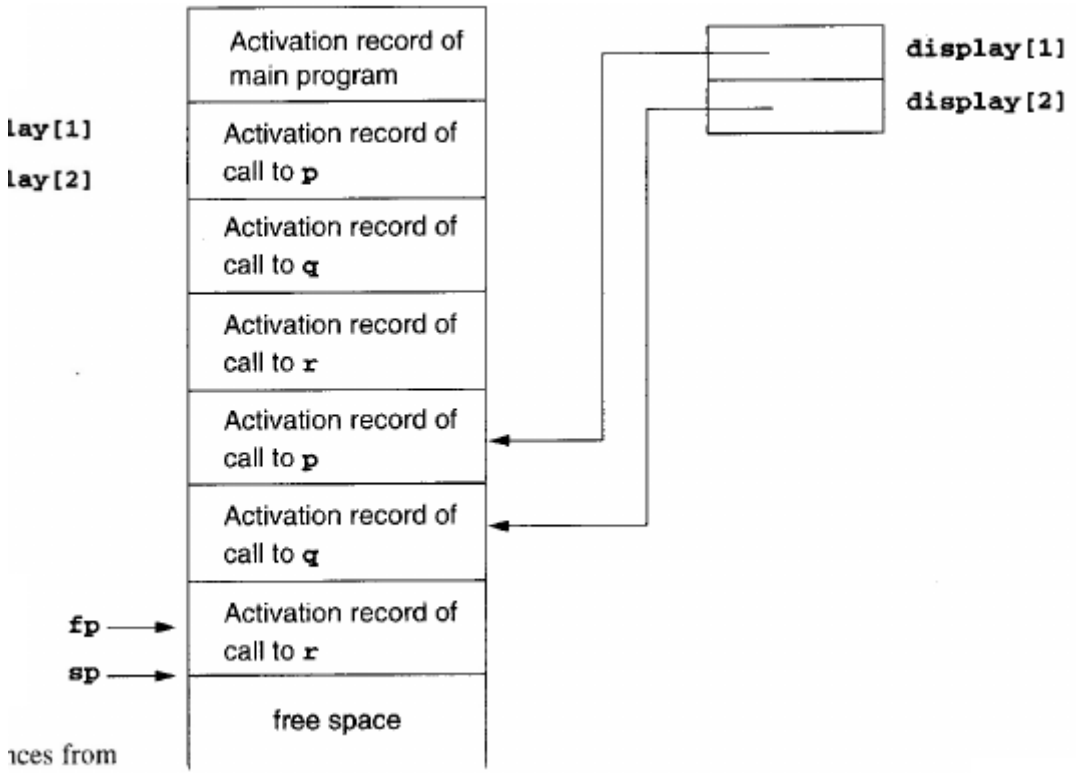

Figure 2: RTE with display (2)

Exercise 4 (Memory layout for classes) [4] Draw the memory layout of objects of the following C++ classes, together with the virtual function tables, as described in [1, Section 7.4.2].

```
1  class A
2  { public:
3    int a;
4    virtual void f();
5    virtual void g();
6  };
7
8  class B : public A
9  { public:
10   int b;
11   virtual void f();
12   void h();
13 };
14
15
16 class C: public B
17 { public:
18   int c;
19   virtual void g();
20 }
```

Exercise 5 (Parameter passing) [5] Give the output of the following program (written in C syntax) using the 4 parameter passing methods discussed in [1, Section 7.5].

[4] The task corresponds to [1, Exercise 7.13.]
[5] The task corresponds to [1, Exercise 7.15.]

```c
#include <stdio.h>
int i = 0;

void p(int x, int y)
{ x += 1;
  i += 1;
  y += 1;
}

main ()
{ int a[2] = {1,1};
  p(a[i],a[i]);
  printf("%d %d\n",a[0], a[1]);
  return 0;
}
```

Exercise 6 (Parameter passing) [6] Give the output of the following program (written in C syntax) using the 4 parameter passing methods discussed in [1, Section 7.5].

```c
#include <stdio.h>
int i = 0;

void swap (int x, int y)
{
  x = x + y;
  y = x - y;
  x = x - y;
}

main ()
{ int a[3] = {1,2,0};
  swap(i,a[i]);
  printf("%d %d %d %d\n",i,a[0], a[1],a[2]);
  return 0;
}
```

References

[1] K. Louden. *Compiler Construction, Principles and Practice.* PWS Publishing, 1997.

[6] The task corresponds to [1, Exercise 7.16.]