CS 333 : Compiler Design
TUTORIAL : Runtime Environments : April 2024

The generic forms discussed in the lecture for calling convention and activation record structure, are given below for reference and are to be used for solving these problems. Activation record (AR) may contain the information required for a single activation of a procedure. However it is possible

1. Local variables 2. Parameters 3. Return address
4. Saved registers 5. Static link 6. Dynamic link 7. Return value

Tasks performed by Caller **(Caller prologue protocol : code inserted by compiler just before a call in the caller's body)**	**Tasks performed by Callee** **(Callee prologue protocol : code inserted by compiler just before the first executable statement in callee's body)**
A1. Make space on the stack for a return value. A2. Pass actual parameters through stack or registers. A3. Sets the static link. A4.Save return address in caller's code on stack. A5. Jump to the called function	C1. Sets the dynamic link. C2. Sets base of new activation record. C3. Saves registers on the stack. C4. Makes space for locals on stack.
Tasks performed by Caller **(Caller epilogue protocol : code inserted by compiler just after a call in the caller's body**	**Tasks performed by Callee** **(Callee epilogue protocol : code inserted by compiler in the callee's body just before it returns control back to its caller)**
B1. Picks return value from stack	D1. Restores registers. D2. Sets base pointer to AR of caller function. D3. Restores Stack pointer to location containing returned value. D4. Retrieves saved return address in caller's code and Returns back to caller

Use the brief description of X86-64 bit assembly given in the lecture notes.

Consider the C program spread out in 2 columns as given below for the two problems that follow.

```
#include <stdio.h>
void f( int x, const int k)
{
  x = x* k;
  if ( x == 0) x = x + k;
  else
    x = x/2;
    printf(" x = %d k = %d \n", x, k);
}
```

```
int main()
{
 int a = 10, b = 2;
 f(a+a*b, b);
 return 0;
}
```

P1. Examine the function f() with respect to the source and its annotated assembly given below in 2 columns.
(a) You have to identify all the callee prologue actions, C1 to C4, that are generally performed by a callee and are present in the given assembly code. You should be able to justify your answer.
(b) Identify all the callee epilogue actions, D1 to D4, that are generally performed by a callee and are present in the given assembly code with brief justifications.
(c) Examine the source code for all names used in its body, {x, k} and how they are referenced in the assembly code. Justify the relevant assembly instructions around the references.
(d) f() is caller for the printf() function. Examine the assembly code that relates to f() as a caller function, and identify the caller prologue and epilogue actions, {A1, .., A5, B1} present in it.

```
void f( int x, const int k)
{


  x = x* k;



  if ( x == 0) x = x + k;



  else


    x = x/2;


    printf(" x = %d k = %d \n", x, k);


}
```

```
.LC0:
        .string  " x = %d k = %d \n"
        .text
        .globl   f
        .type    f, @function
f:
        pushq    %rbp    #
        movq     %rsp, %rbp      #,
        subq     $16, %rsp       #,
        movl     %edi, -4(%rbp)  # x, x
        movl     %esi, -8(%rbp)  # k, k
        movl     -4(%rbp), %eax  # x, tmp88
        imull    -8(%rbp), %eax  # k, tmp87
        movl     %eax, -4(%rbp)  # tmp87, x
        cmpl     $0, -4(%rbp)    #, x
        jne      .L2     #,
        movl     -8(%rbp), %eax  # k, tmp89
        addl     %eax, -4(%rbp)  # tmp89, x
        jmp      .L3     #
.L2:
        movl     -4(%rbp), %eax  # x, tmp91
        movl     %eax, %edx      # tmp91, tmp92
        shrl     $31, %edx       #, tmp92
        addl     %edx, %eax      # tmp92, tmp93
        sarl     %eax    # tmp94
        movl     %eax, -4(%rbp)  # tmp94, x
.L3:
        movl     -8(%rbp), %edx  # k, tmp95
        movl     -4(%rbp), %eax  # x, tmp96
        movl     %eax, %esi      # tmp96,
        movl     $.LC0, %edi     #,
        movl     $0, %eax        #,
        call     printf  #
        nop
        leave
        ret
```

P2. Examine the function main() with respect to the source and its annotated assembly given below in 2 columns.

(a) You have to identify all the callee prologue and epilogue actions, C1 to C4 and D1 to D4, that are performed by main() as a callee in the assembly code given below with justifications.

(b) Identify all the caller prologue and epilogue actions , A1 to A4, B1 performed by main() as a caller of f() in its body with brief justifications.

(c) Examine the source code for all names used in its body, {a, b} and show how they are referenced in the assembly code. Justify the relevant assembly instructions around the references.

```
int main()

{


  int a = 10, b = 2;


  f(a+a*b, b);


  return 0;



}
```

```
        .globl  main
        .type   main, @function
main:
        pushq   %rbp            #
        movq    %rsp, %rbp      #,
        subq    $16, %rsp       #,
        movl    $10, -8(%rbp)   #, a
        movl    $2, -4(%rbp)    #, b
        movl    -4(%rbp), %eax  # b, tmp91
        addl    $1, %eax        #, D.2302
        imull   -8(%rbp), %eax  # a, D.2302
        movl    -4(%rbp), %edx  # b, tmp92
        movl    %edx, %esi      # tmp92,
        movl    %eax, %edi      # D.2302,
        call    f               #
        movl    $0, %eax        #, D.2302
        leave
        ret
```

Examine the source C program given below and answer the two problems that follow.

```
int f(int x1, int x2, int x3, int*p1, int x4, int x5,
      int x6, int*p2)
{
  int a, b, c[100];
  a = *p1 + x1 + x2 + x3 ;
  b = *p2 + x4 + x5 + x6;
  c[0]= 1; c[99] = 5;
  return a+b;
}
```

```
int main()
{
  int i1 = 1, i2 = 2, i3 = 3, i4 = 4, i5 = 5, i6 = 6, res;
  int *q1 = &i1, *q2 = &i2;
  res = f(i1, i2, i3, q1, i4, i5, i6, &i2);
  return 6;
}
```

P3. Examine the function f() with respect to the source and its annotated assembly given below in 2 columns.
(a) You have to identify all the callee prologue and epilogue actions, C1 to C4 and D1 to D4, that are performed by f() as evidenced in the given assembly code. Justify your answer briefly.
(b) Examine the source code for all names used in the body of f() and explain why they are referenced in the assembly code in the manner that is done by gcc. Justify the relevant assembly instructions around the references.
(c) Identify in the assembly code of f(), the instructions that relate to parameter passing mechanisms, correlate with the corresponding source code. Justify the appropriateness of the code generated by gcc for this purpose.

```
int f(int x1, int x2, int x3, int*p1, int x4, int x5,
      int x6, int*p2)
{
  int a, b, c[100];
  a = *p1 + x1 + x2 + x3 ;
  b = *p2 + x4 + x5 + x6;
  c[0]= 1; c[99] = 5;
  return a+b;
}
```

```
        .globl  f
        .type   f, @function
f:
        pushq   %rbp    #
        movq    %rsp, %rbp      #,
        subq    $480, %rsp      #,
        movl    %edi, -436(%rbp)        # x1, x1
        movl    %esi, -440(%rbp)        # x2, x2
        movl    %edx, -444(%rbp)        # x3, x3
        movq    %rcx, -456(%rbp)        # p1, p1
        movl    %r8d, -448(%rbp)        # x4, x4
        movl    %r9d, -460(%rbp)        # x5, x5
        movq    24(%rbp), %rax  # p2, tmp95
        movq    %rax, -472(%rbp)        # tmp95, p2
        movq    %fs:40, %rax    #, tmp113
        movq    %rax, -8(%rbp)  # tmp113, D.2325
        xorl    %eax, %eax      # tmp113
        movq    -456(%rbp), %rax        # p1, tmp96
        movl    (%rax), %edx    # *p1_2(D), D.2324
        movl    -436(%rbp), %eax        # x1, tmp97
        addl    %eax, %edx      # tmp97, D.2324
        movl    -440(%rbp), %eax        # x2, tmp98
```

```
addl    %eax, %edx      # tmp98, D.2324
movl    -444(%rbp), %eax        # x3, tmp102
addl    %edx, %eax      # D.2324, tmp101
movl    %eax, -424(%rbp)        # tmp101, a
movq    -472(%rbp), %rax        # p2, tmp103
movl    (%rax), %edx    # *p2_10(D), D.2324
movl    -448(%rbp), %eax        # x4, tmp104
addl    %eax, %edx      # tmp104, D.2324
movl    -460(%rbp), %eax        # x5, tmp105
addl    %eax, %edx      # tmp105, D.2324
movl    16(%rbp), %eax # x6, tmp109
addl    %edx, %eax      # D.2324, tmp108
movl    %eax, -420(%rbp)        # tmp108, b
movl    $1, -416(%rbp)  #, c
movl    $5, -20(%rbp)   #, c
movl    -424(%rbp), %edx        # a, tmp110
movl    -420(%rbp), %eax        # b, tmp111
addl    %edx, %eax      # tmp110, D.2324
movq    -8(%rbp), %rcx  # D.2325, tmp114
xorq    %fs:40, %rcx    #, tmp114
leave
ret
```

P4. Examine the function main() with respect to the source and its annotated assembly given below in 2 columns.
(a) You have to identify all the callee prologue and epilogue actions, C1 to C4 and D1 to D4, that are performed by main() as a callee in the assembly code given below with justifications.
(b) Identify all the caller prologue and epilogue actions , A1 to A4, B1 performed by main() as a caller of f() in its body with brief justifications.
(c) Examine the source code for all names used in its body, and show how they are referenced in the assembly code. Justify the relevant assembly instructions around the references.
(d) Identify in the assembly code of main(), the instructions that relate to parameter passing mechanisms, correlate with the corresponding source code. Justify the appropriateness of the code generated by gcc for this purpose.

```
int main()
{
  int i1 = 1, i2 = 2, i3 = 3, i4 = 4, i5 = 5, i6 = 6, res;
  int *q1 = &i1, *q2 = &i2;
  res = f(i1, i2, i3, q1, i4, i5, i6, &i2);
  return 6;
}
```

```
        .globl  main
        .type   main, @function
main:
        pushq   %rbp    #
        movq    %rsp, %rbp      #,
        subq    $64, %rsp       #,
        movq    %fs:40, %rax    #, tmp101
        movq    %rax, -8(%rbp)  # tmp101, D.2329
        xorl    %eax, %eax      # tmp101
        movl    $1, -52(%rbp)   #, i1
        movl    $2, -48(%rbp)   #, i2
        movl    $3, -44(%rbp)   #, i3
        movl    $4, -40(%rbp)   #, i4
        movl    $5, -36(%rbp)   #, i5
        movl    $6, -32(%rbp)   #, i6
        leaq    -52(%rbp), %rax         #, tmp91
        movq    %rax, -24(%rbp)         # tmp91, q1
        leaq    -48(%rbp), %rax         #, tmp92
        movq    %rax, -16(%rbp)         # tmp92, q2
        movl    -48(%rbp), %esi # i2, D.2328
        movl    -52(%rbp), %eax         # i1, D.2328
        movl    -36(%rbp), %r9d         # i5, tmp93
        movl    -40(%rbp), %r8d         # i4, tmp94
        movq    -24(%rbp), %rcx         # q1, tmp95
        movl    -44(%rbp), %edx         # i3, tmp96
        leaq    -48(%rbp), %rdi #, tmp97
        pushq   %rdi    # tmp97
        movl    -32(%rbp), %edi# i6, tmp98
        pushq   %rdi    # tmp98
        movl    %eax, %edi      # D.2328,
        call    f       #
        addq    $16, %rsp       #,
        movl    %eax, -28(%rbp)         # tmp99, res
        movl    $6, %eax        #, D.2328
        movq    -8(%rbp), %rdx  # D.2329, tmp102
        xorq    %fs:40, %rdx    #, tmp102
        leave
        ret
```

End of Tutorial Problems