An Example Language Program D

The following code shows an example C language program; the second example code shows the corresponding assembly code generated by SPARCompiler C 3.0.2 that runs on the Solaris 2.*x* operating environment. Comments have been added to the asembly code to show correspondence to the C code.

The following C Program computes the first *n* Fibonacci numbers:

Figure D-1 C Program Example Source

```
/* a simple program computing the first n Fibonacci numbers */

extern unsigned * fibonacci();

#define MAX_FIB_REPRESENTABLE 49


/* compute the first n Fibonacci numbers */
unsigned * fibonacci(n)
     int n;
{
  static unsigned fib_array[MAX_FIB_REPRESENTABLE] = {0,1};
  unsigned prev_number = 0;
  unsigned curr_number = 1;
  int i;

  if (n >= MAX_FIB_REPRESENTABLE) {
    printf("Fibonacci(%d) cannot be represented in a 32 bit word\n", n);
    exit(1);
  }

  for (i = 2; i < n; i++) {
    fib_array[i] = prev_number + curr_number;
    prev_number = curr_number;
    curr_number = fib_array[i];
  }

  return(fib_array);
}


main()
{
  int n, i;
  unsigned * result;

  printf("Fibonacci(n):, please enter n:\n");
  scanf("%d", &n);

  result = fibonacci(n);
  for (i = 1; i <= n; i++)
    printf("Fibonacci (%d) is %u\n", i, *result++);
}
```

The C SPARCompiler generates the following assembler output for the Fibonacci number C source. Annotation has been added to help you understand the code.

Figure D-2 Assembler Output From C Source

```
!
!  a simple program computing the first n Fibonacci numbers,
!  showing various pseudo-operations, sparc instructions, synthetic instructions
!
!  pseudo-operations:  .align, .ascii, .file, .global, .ident, .proc, .section,
!                                                .size, .skip, .type, .word
!  sparc instructions:    add, bg, bge, bl, ble, ld, or, restore, save, sethi, st
!  synthetic instructions: call, cmp, inc, mov, ret
!

   .file          "fibonacci.c"             ! the original source file name


   .section       ".text"                   ! text section (executable instructions)
   .proc          79                        ! subroutine fibonacci, it's return
                                            ! value will be in %i0
   .global        fibonacci                 ! fibonacci() can be referenced
                                            ! outside this file
   .align         4                         ! align the beginning of this section
                                            ! to word boundary
fibonacci:
   save           %sp,-96,%sp               ! create new stack frame and register
                                            ! window for this subroutine
/*  if (n >= MAX_FIB_REPRESENTABLE) { */
                                            ! note, C style comment strings are
                                            ! also permitted
   cmp            %i0,49                    ! n >= MAX_FIB_REPRESENTABLE ?
                                            ! note, n, the 1st parameter to
                                            ! fibonacci(), is stored in %i0 upon
                                            ! entry
   bl             .L77003
   mov            0,%i2                     ! initialization of variable
                                            ! prev_number is executed in the
                                            ! delay slot

/* printf("Fibonacci(%d) cannot be represented in a 32 bits word\n", n); */
   sethi          %hi(.L20),%o0             ! if branch not taken, call printf(),
   or             %o0,%lo(.L20),%o0         ! set up 1st, 2nd argument in %o0, %o1;
   call           printf,2                  ! the ",2" means there are 2 out
   mov            %i0,%o1                   ! registers used as arguments
/* exit(1); */
   call           exit,1
   mov            1,%o0
.L77003:                                    ! initialize variables before the loop
/* for (i = 2; i < n; i++) { */
   mov            1,%i4                     ! curr_number = 1
```

```
   mov           2,%i3                     ! i = 2
   cmp           %i3,%i0                   ! i <= n?
   bge           .L77006                   ! if not, return
   sethi         %hi(.L16+8),%o0           ! use %i5 to store &fib_array[i]
   add           %o0,%lo(.L16+8),%i5
.LY1:                                      ! loop body
/* fib_array[i] = prev_number + curr_number; */
   add           %i2,%i4,%i2               ! fib_array[i] = prev_number+curr_number
   st            %i2,[%i5]
/* prev_number = curr_number; */
   mov           %i4,%i2                   ! prev_number = curr_number
/* curr_number = fib_array[i]; */
   ld            [%i5],%i4                 ! curr_number = fib_array[i]
   inc           %i3                       ! i++
   cmp           %i3,%i0                   ! i <= n?
   bl            .LY1                      ! if yes, repeat loop
   inc           4,%i5                     ! increment ptr to fib_array[]
.L77006:
/* return(fib_array); */
   sethi         %hi(.L16),%o0             ! return fib_array in %i0
   add           %o0,%lo(.L16),%i0
   ret
   restore                                 ! destroy stack frame and register
                                           ! window
   .type         fibonacci,#function       ! fibonacci() is of type function
   .size         fibonacci,(.-fibonacci)   ! size of function:
                                           !  current location counter minus
                                           !  beginning definition of function


   .proc         18                        ! main program
   .global       main
   .align        4
main:
   save          %sp,-104,%sp              ! create stack frame for main()
/* printf("Fibonacci(n):, please input n:\n"); */
   sethi         %hi(.L31),%o0             ! call printf, with 1st arg in %o0
   call          printf,1
   or            %o0,%lo(.L31),%o0
/* scanf("%d", &n); */
   sethi         %hi(.L33),%o0             ! call scanf, with 1st arg, in %o0
   or            %o0,%lo(.L33),%o0         ! move 2nd arg. to %o1, in delay slot
   call          scanf,2
   add           %fp,-4,%o1

/* result = fibonacci(n); */
   call          fibonacci,1
   ld            [%fp-4],%o0

                                           ! some initializations before the for-
                                           ! loop, put the variables in registers
/* for (i = 1; i <= n; i++) */
   mov           1,%i5                     ! %i5 <-- i
   mov           %o0,%i4                   ! %i4 <-- result
```

```
    sethi           %hi(.L38),%o0             ! %i2 <-- format string for printf
    add             %o0,%lo(.L38),%i2
    ld              [%fp-4],%o0               ! test if (i <= n) ?
    cmp             %i5,%o0                   ! note, n is stored in [%fp-4]
    bg              .LE27
    nop
.LY2:                                         ! loop body
/* printf("Fibonacci (%d) is %u\n", i, *result++); */
    ld              [%i4],%o2                 ! call printf, with (*result) in %o2,
    mov             %i5,%o1                   !   i in %o1, format string in %o0
    call            printf,3
    mov             %i2,%o0
    inc             %i5                       ! i++
    ld              [%fp-4],%o0               ! i <= n?
    cmp             %i5,%o0
    ble             .LY2
    inc             4,%i4                     ! result++
.LE27:
    ret
    restore
    .type           main,#function            ! type and size of main
    .size           main,(.-main)


    .section ".data"                          ! switch to data section
                                              ! (contains initialized data)
    .align          4
.L16:
/* static unsigned fib_array[MAX_FIB_REPRESENTABLE] = {0,1}; */
    .align          4                         ! initialization of first 2 elements
    .word           0                         ! of fib_array[]
    .align          4
    .word           1
    .skip           188
    .type           .L16,#object              ! storage allocation for the rest of
                                              ! fib_array[]


    .section ".data1"                         ! the ascii string data are entered
                                              ! into the .data1 section;
                                              ! #alloc:  memory would be allocated
                                              !    for this section during run time
                                              ! #write:  the section contains data
                                              !    that is writeable during process
                                              !    execution
    .align          4
.L20:                                         ! ascii strings used in the printf stmts
    .ascii          "Fibonacci(%d) cannot be represented in a 32 bit w"
    .ascii          "ord\n\0"
    .align          4                         ! align the next ascii string to word
                                              ! boundary
.L31:
    .ascii          "Fibonacci(n):, please enter n:\n\0"
    .align          4
```

```
.L33:
    .ascii          "%d\0"
    .align          4
.L38:
    .ascii          "Fibonacci (%d) is %u\n\0"
    .ident          "acomp: (CDS) SPARCompilers 2.0 05 Jun 1991"
                                         ! an idenitfication string produced
                                         ! by the compiler to be entered into
                                         ! the .comment section
```