

Princeton University

**COS 217: Introduction to Programming Systems**  
**The ARMv8 Function Call Conventions**

When  $f()$  calls  $g()$  ...

**Rule 1**

**Q:** Where will  $f()$  place the return **address** and where will  $g()$  find that return **address**?

**A:** In register X30. That is, the `bl` instruction places the address of the instruction following the `bl` instruction in register X30, and the `ret` instruction branches to the address in register X30.

**Rule 2**

**Q:** Where will  $f()$  place its arguments and where will  $g()$  find its parameters?

**A:** In registers R0...R7 in that order.

**Rule 3**

**Q:** Where will  $g()$  place its return **value** and where will  $f()$  find that return **value**?

**A:** In register R0.

**Rule 4**

**Q:** Which registers may  $g()$  affect?

**A:** **Callee-saved** registers (informally, the **g-saved** registers): **R19...R28**

The callee/g **may not** change the contents of those registers.

The callee/g must either:

Not change the contents of those registers, or

Save the contents of those registers before it changes them, and restore the contents before it returns – thus giving the caller/f the illusion that the contents of those registers were not changed.

**Caller-saved** registers (informally, the **f-saved** registers): **R0...R7, R9...R15**

The callee/g **may** change the contents of those registers.

If the caller/f requires that the contents of those registers be preserved across its call of the callee/g, then the caller/f must do the preserving:

The caller/f must save the contents of those registers before calling the callee/g.

The caller/f must restore the old contents of those registers after calling the callee/g.

## euclidglobal.c (Page 1 of 1)

```

1: /*-----*/
2: /* euclidglobal.c */
3: /* Author: Bob Dondero */
4: /*-----*/
5:
6: #include <stdio.h>
7: #include <stdlib.h>
8:
9: /*-----*/
10:
11: static long l1; /* Bad style. */
12: static long l2; /* Bad style. */
13: static long lGcd; /* Bad style. */
14: static long lTemp; /* Bad style. */
15: static long lAbsFirst; /* Bad style. */
16: static long lAbsSecond; /* Bad style. */
17:
18: /*-----*/
19:
20: /* Assign to lGcd the greatest common divisor of l1 and l2. */
21:
22: static void gcd(void)
23: {
24:     lAbsFirst = labs(l1);
25:     lAbsSecond = labs(l2);
26:
27:     while (lAbsSecond != 0)
28:     {
29:         lTemp = lAbsFirst % lAbsSecond;
30:         lAbsFirst = lAbsSecond;
31:         lAbsSecond = lTemp;
32:     }
33:
34:     lGcd = lAbsFirst;
35: }
36:
37: /*-----*/
38:
39: /* Read two integers from stdin. Compute their greatest common divisor,
40: and write it to stdout. Return 0. */
41:
42: int main(void)
43: {
44:     printf("Enter an integer: ");
45:     scanf("%ld", &l1); /* Should validate. */
46:
47:     printf("Enter an integer: ");
48:     scanf("%ld", &l2); /* Should validate. */
49:
50:     gcd();
51:
52:     printf("The gcd is %ld\n", lGcd);
53:
54:     return 0;
55: }

```

## euclidglobalflat.c (Page 1 of 1)

```

1: /*-----*/
2: /* euclidglobalflat.c */
3: /* Author: Bob Dondero */
4: /*-----*/
5:
6: #include <stdio.h>
7: #include <stdlib.h>
8:
9: /*-----*/
10:
11: static long l1; /* Bad style. */
12: static long l2; /* Bad style. */
13: static long lGcd; /* Bad style. */
14: static long lTemp; /* Bad style. */
15: static long lAbsFirst; /* Bad style. */
16: static long lAbsSecond; /* Bad style. */
17:
18: /*-----*/
19:
20: /* Assign to lGcd the greatest common divisor of l1 and l2. */
21:
22: static void gcd(void)
23: {
24:     lAbsFirst = labs(l1);
25:     lAbsSecond = labs(l2);
26:     gcdLoop:
27:     if (lAbsSecond == 0) goto gcdLoopEnd;
28:     lTemp = lAbsFirst % lAbsSecond;
29:     lAbsFirst = lAbsSecond;
30:     lAbsSecond = lTemp;
31:     goto gcdLoop;
32: gcdLoopEnd:
33:     lGcd = lAbsFirst;
34: }
35:
36: /*-----*/
37:
38: /* Read two integers from stdin. Compute their greatest common divisor,
39: and write it to stdout. Return 0. */
40:
41: int main(void)
42: {
43:     printf("Enter an integer: ");
44:     scanf("%ld", &l1); /* Should validate. */
45:
46:     printf("Enter an integer: ");
47:     scanf("%ld", &l2); /* Should validate. */
48:
49:     gcd();
50:
51:     printf("The gcd is %ld\n", lGcd);
52:
53:     return 0;
54: }

```

euclidglobal.s (Page 1 of 3)

```

1: //-----
2: // euclidglobal.s
3: // Author: Bob Dondero
4: //-----
5:
6:     .section .rodata
7:
8: promptStr:
9:     .string "Enter an integer: "
10:
11: scanfFormatStr:
12:     .string "%ld"
13:
14: printfFormatStr:
15:     .string "The gcd is %ld\n"
16:
17: //-----
18:
19:     .section .data
20:
21: //-----
22:
23:     .section .bss
24:
25: l1:
26:     .skip 8
27: l2:
28:     .skip 8
29: lGcd:
30:     .skip 8
31: lTemp:
32:     .skip 8
33: lAbsFirst:
34:     .skip 8
35: lAbsSecond:
36:     .skip 8
37:
38: //-----
39:
40:     .section .text
41:
42: //-----
43: // Assign to lGcd the greatest common divisor of l1 and l2.
44: // void gcd(void)
45: //-----
46:
47: // Must be a multiple of 16
48: .equ GCD_STACK_BYTECOUNT, 16
49:
50: gcd:
51:
52: // Prolog
53: sub    sp, sp, GCD_STACK_BYTECOUNT
54: str    x30, [sp]
55:
56: // lAbsFirst = labs(l1)
57: adr    x0, l1
58: ldr    x0, [x0]
59: bl     labs
60: adr    x1, lAbsFirst
61: str    x0, [x1]
62:
63: // lAbsSecond = labs(l1)

```

euclidglobal.s (Page 2 of 3)

```

64:     adr    x0, l2
65:     ldr    x0, [x0]
66:     bl     labs
67:     adr    x1, lAbsSecond
68:     str    x0, [x1]
69:
70: gcdLoop:
71:
72:     // if (lAbsSecond == 0) goto gcdLoopEnd
73:     adr    x0, lAbsSecond
74:     ldr    x0, [x0]
75:     cmp    x0, 0
76:     beq    gcdLoopEnd
77:
78:     // lTemp = lAbsFirst % lAbsSecond
79:     // remainder = (dividend - (quotient * divisor))
80:     adr    x0, lAbsFirst
81:     ldr    x0, [x0]
82:     adr    x1, lAbsSecond
83:     ldr    x1, [x1]
84:     sdiv   x2, x0, x1
85:     mul    x3, x2, x1
86:     sub    x4, x0, x3
87:     adr    x0, lTemp
88:     str    x4, [x0]
89:
90:     // lAbsFirst = lAbsSecond
91:     adr    x0, lAbsSecond
92:     ldr    x0, [x0]
93:     adr    x1, lAbsFirst
94:     str    x0, [x1]
95:
96:     // lAbsSecond = lTemp
97:     adr    x0, lTemp
98:     ldr    x0, [x0]
99:     adr    x1, lAbsSecond
100:    str    x0, [x1]
101:
102:    // goto gcdLoop
103:    b      gcdLoop
104:
105: gcdLoopEnd:
106:
107:    // lGcd = lAbsFirst
108:    adr    x0, lAbsFirst
109:    ldr    x0, [x0]
110:    adr    x1, lGcd
111:    str    x0, [x1]
112:
113:    // Epilog and return
114:    ldr    x30, [sp]
115:    add    sp, sp, GCD_STACK_BYTECOUNT
116:    ret
117:
118:    .size  gcd, (. - gcd)
119:
120: //-----
121: // Read two integers from stdin. Compute their greatest common
122: // divisor, and write it to stdout. Return 0.
123: // int main(void)
124: //-----
125:
126: // Must be a multiple of 16

```



## euclidglobal.s (Page 3 of 3)

```
127:      .equ   MAIN_STACK_BYTECOUNT, 16
128:
129:      .global main
130:
131: main:
132:
133:      // Prolog
134:      sub   sp, sp, MAIN_STACK_BYTECOUNT
135:      str   x30, [sp]
136:
137:      // printf("Enter an integer: ")
138:      adr   x0, promptStr
139:      bl   printf
140:
141:      // scanf("%ld", &l1)
142:      adr   x0, scanfFormatStr
143:      adr   x1, l1
144:      bl   scanf
145:
146:      // printf("Enter an integer: ")
147:      adr   x0, promptStr
148:      bl   printf
149:
150:      // scanf("%ld", &l2)
151:      adr   x0, scanfFormatStr
152:      adr   x1, l2
153:      bl   scanf
154:
155:      // gcd()
156:      bl   gcd
157:
158:      // printf("The gcd is %ld\n", lGcd)
159:      adr   x0, printfFormatStr
160:      adr   x1, lGcd
161:      ldr   x1, [x1]
162:      bl   printf
163:
164:      // Epilog and return 0
165:      mov   w0, 0
166:      ldr   x30, [sp]
167:      add   sp, sp, MAIN_STACK_BYTECOUNT
168:      ret
169:
170:      .size main, (. - main)
```

## euclid.c (Page 1 of 1)

```

1: /*-----*/
2: /* euclid.c */
3: /* Author: Bob Dondero */
4: /*-----*/
5:
6: #include <stdio.h>
7: #include <stdlib.h>
8:
9: /*-----*/
10:
11: /* Return the greatest common divisor of lFirst and lSecond. */
12:
13: static long gcd(long lFirst, long lSecond)
14: {
15:     long lTemp;
16:     long lAbsFirst;
17:     long lAbsSecond;
18:
19:     lAbsFirst = labs(lFirst);
20:     lAbsSecond = labs(lSecond);
21:
22:     while (lAbsSecond != 0)
23:     {
24:         lTemp = lAbsFirst % lAbsSecond;
25:         lAbsFirst = lAbsSecond;
26:         lAbsSecond = lTemp;
27:     }
28:
29:     return lAbsFirst;
30: }
31:
32: /*-----*/
33:
34: /* Read two integers from stdin. Compute their greatest common divisor,
35:    and write it to stdout. Return 0. */
36:
37: int main(void)
38: {
39:     long l1;
40:     long l2;
41:     long lGcd;
42:
43:     printf("Enter an integer: ");
44:     scanf("%ld", &l1); /* Should validate. */
45:
46:     printf("Enter an integer: ");
47:     scanf("%ld", &l2); /* Should validate. */
48:
49:     lGcd = gcd(l1, l2);
50:
51:     printf("The gcd is %ld\n", lGcd);
52:
53:     return 0;
54: }

```

## euclidflat.c (Page 1 of 1)

```

1: /*-----*/
2: /* euclidflat.c */
3: /* Author: Bob Dondero */
4: /*-----*/
5:
6: #include <stdio.h>
7: #include <stdlib.h>
8:
9: /*-----*/
10:
11: /* Return the greatest common divisor of lFirst and lSecond. */
12:
13: static long gcd(long lFirst, long lSecond)
14: {
15:     long lTemp;
16:     long lAbsFirst;
17:     long lAbsSecond;
18:
19:     lAbsFirst = labs(lFirst);
20:     lAbsSecond = labs(lSecond);
21:
22: gcdLoop:
23:     if (lAbsSecond == 0) goto gcdLoopEnd;
24:     lTemp = lAbsFirst % lAbsSecond;
25:     lAbsFirst = lAbsSecond;
26:     lAbsSecond = lTemp;
27:     goto gcdLoop;
28: gcdLoopEnd:
29:     return lAbsFirst;
30: }
31:
32: /*-----*/
33:
34: /* Read two integers from stdin. Compute their greatest common divisor,
35:    and write it to stdout. Return 0. */
36:
37: int main(void)
38: {
39:     long l1;
40:     long l2;
41:     long lGcd;
42:
43:     printf("Enter an integer: ");
44:     scanf("%ld", &l1); /* Should validate. */
45:
46:     printf("Enter an integer: ");
47:     scanf("%ld", &l2); /* Should validate. */
48:
49:     lGcd = gcd(l1, l2);
50:
51:     printf("The gcd is %ld\n", lGcd);
52:
53:     return 0;
54: }

```

euclid.s (Page 1 of 3)

```

1: //-----
2: // euclid.s
3: // Author: Bob Dondero
4: //-----
5:
6:     .section .rodata
7:
8: promptStr:
9:     .string "Enter an integer: "
10:
11: scanfFormatStr:
12:     .string "%ld"
13:
14: printfFormatStr:
15:     .string "The gcd is %ld\n"
16:
17: //-----
18:
19:     .section .data
20:
21: //-----
22:
23:     .section .bss
24:
25: //-----
26:
27:     .section .text
28:
29: //-----
30: // Return the greatest common divisor of lFirst and lSecond.
31: // long gcd(long lFirst, long lSecond)
32: //-----
33:
34: // Must be a multiple of 16
35: .equ GCD_STACK_BYTECOUNT, 48
36:
37: // Local variable stack offsets:
38: .equ LABSSECOND, 8
39: .equ LABSFIRST, 16
40: .equ LTEMP, 24
41:
42: // Parameter stack offsets:
43: .equ LSECOND, 32
44: .equ LFIRST, 40
45:
46: gcd:
47:
48: // Prolog
49: sub sp, sp, GCD_STACK_BYTECOUNT
50: str x30, [sp]
51: str x0, [sp, LFIRST]
52: str x1, [sp, LSECOND]
53:
54: // long lTemp
55: // long lAbsFirst
56: // long lAbsSecond
57:
58: // lAbsFirst = labs(lFirst)
59: ldr x0, [sp, LFIRST]
60: bl labs
61: str x0, [sp, LABSFIRST]
62:
63: // lAbsSecond = labs(lSecond)

```

euclid.s (Page 2 of 3)

```

64:     ldr x0, [sp, LSECOND]
65:     bl labs
66:     str x0, [sp, LABSSECOND]
67:
68: gcdLoop:
69:
70:     // if (lAbsSecond == 0) goto gcdLoopEnd
71:     ldr x0, [sp, LABSSECOND]
72:     cmp x0, 0
73:     beq gcdLoopEnd
74:
75:     // lTemp = lAbsFirst % lAbsSecond
76:     // remainder = (dividend - (quotient * divisor))
77:     ldr x0, [sp, LABSFIRST]
78:     ldr x1, [sp, LABSSECOND]
79:     sdiv x2, x0, x1
80:     mul x3, x2, x1
81:     sub x4, x0, x3
82:     str x4, [sp, LTEMP]
83:
84:     // lAbsFirst = lAbsSecond
85:     ldr x0, [sp, LABSSECOND]
86:     str x0, [sp, LABSFIRST]
87:
88:     // lAbsSecond = lTemp
89:     ldr x0, [sp, LTEMP]
90:     str x0, [sp, LABSSECOND]
91:
92:     // goto gcdLoop
93:     b gcdLoop
94:
95: gcdLoopEnd:
96:
97: // Epilog and return lAbsFirst
98: ldr x0, [sp, LABSFIRST]
99: ldr x30, [sp]
100: add sp, sp, GCD_STACK_BYTECOUNT
101: ret
102:
103: .size gcd, (. - gcd)
104:
105: //-----
106: // Read two integers from stdin. Compute their greatest common
107: // divisor, and write it to stdout. Return 0.
108: // int main(void)
109: //-----
110:
111: // Must be a multiple of 16
112: .equ MAIN_STACK_BYTECOUNT, 32
113:
114: // Local variables stack offsets:
115: .equ LGCD, 8
116: .equ L2, 16
117: .equ L1, 24
118:
119: .global main
120:
121: main:
122:
123: // Prolog
124: sub sp, sp, MAIN_STACK_BYTECOUNT
125: str x30, [sp]
126:

```

euclid.s (Page 3 of 3)

```
127:      // long l1
128:      // long l2
129:      // long lGcd
130:
131:      // printf("Enter an integer: ")
132:      adr    x0, promptStr
133:      bl     printf
134:
135:      // scanf("%ld", &l1)
136:      adr    x0, scanfFormatStr
137:      add    x1, sp, L1
138:      bl     scanf
139:
140:      // printf("Enter an integer: ")
141:      adr    x0, promptStr
142:      bl     printf
143:
144:      // scanf("%ld", &l2)
145:      adr    x0, scanfFormatStr
146:      add    x1, sp, L2
147:      bl     scanf
148:
149:      // lGcd = gcd(l1, l2)
150:      ldr    x0, [sp, L1]
151:      ldr    x1, [sp, L2]
152:      bl     gcd
153:      str    x0, [sp, LGCD]
154:
155:      // printf("The gcd is %ld\n", lGcd)
156:      adr    x0, printfFormatStr
157:      ldr    x1, [sp, LGCD]
158:      bl     printf
159:
160:      // Epilog and return 0
161:      mov    w0, 0
162:      ldr    x30, [sp]
163:      add    sp, sp, MAIN_STACK_BYTECOUNT
164:      ret
165:
166:      .size main, (. - main)
```

euclidopt.s (Page 1 of 3)

```

1: //-----
2: // euclidopt.s
3: // Author: Bob Dondero
4: //-----
5:
6:     .section .rodata
7:
8: promptStr:
9:     .string "Enter an integer: "
10:
11: scanfFormatStr:
12:     .string "%ld"
13:
14: printfFormatStr:
15:     .string "The gcd is %ld\n"
16:
17: //-----
18:
19:     .section .data
20:
21: //-----
22:
23:     .section .bss
24:
25: //-----
26:
27:     .section .text
28:
29:     //-----
30:     // Return the greatest common divisor of lFirst and lSecond.
31:     // long gcd(long lFirst, long lSecond)
32:     //-----
33:
34:     // Must be a multiple of 16
35:     .equ GCD_STACK_BYTECOUNT, 48
36:
37:     // Local variable registers:
38:     LABSSECOND .req x23 // Callee-saved
39:     LABSFIRST .req x22 // Callee-saved
40:     LTEMP .req x21 // Callee-saved
41:
42:     // Parameter registers:
43:     LSECOND .req x20 // Callee-saved
44:     LFIRST .req x19 // Callee-saved
45:
46: gcd:
47:
48:     // Prolog
49:     sub sp, sp, GCD_STACK_BYTECOUNT
50:     str x30, [sp]
51:     str x19, [sp, 8]
52:     str x20, [sp, 16]
53:     str x21, [sp, 24]
54:     str x22, [sp, 32]
55:     str x23, [sp, 40]
56:
57:     // Store parameters in registers
58:     mov LFIRST, x0
59:     mov LSECOND, x1
60:
61:     // long lTemp
62:     // long lAbsFirst
63:     // long lAbsSecond

```

euclidopt.s (Page 2 of 3)

```

64:
65:     // lAbsFirst = labs(lFirst)
66:     mov x0, LFIRST // unnecessary
67:     bl labs
68:     mov LABSFIRST, x0
69:
70:     // lAbsSecond = labs(lSecond)
71:     mov x0, LSECOND
72:     bl labs
73:     mov LABSSECOND, x0
74:
75: gcdLoop:
76:
77:     // if (lAbsSecond == 0) goto gcdLoopEnd
78:     cmp LABSSECOND, 0
79:     beq gcdLoopEnd
80:
81:     // lTemp = lAbsFirst % lAbsSecond
82:     // remainder = (dividend - (quotient * divisor))
83:     sdiv LTEMP, LABSFIRST, LABSSECOND
84:     mul x3, LTEMP, LABSSECOND
85:     sub LTEMP, LABSFIRST, x3
86:
87:     // lAbsFirst = lAbsSecond
88:     mov LABSFIRST, LABSSECOND
89:
90:     // lAbsSecond = lTemp
91:     mov LABSSECOND, LTEMP
92:
93:     // goto gcdLoop
94:     b gcdLoop
95:
96: gcdLoopEnd:
97:
98:     // Epilog and return lAbsFirst
99:     mov x0, LABSFIRST
100:    ldr x30, [sp]
101:    ldr x19, [sp, 8]
102:    ldr x20, [sp, 16]
103:    ldr x21, [sp, 24]
104:    ldr x22, [sp, 32]
105:    ldr x23, [sp, 40]
106:    add sp, sp, GCD_STACK_BYTECOUNT
107:    ret
108:
109:    .size gcd, (. - gcd)
110:
111:    // -----
112:    // Read two integers from stdin. Compute their greatest common
113:    // divisor, and write it to stdout. Return 0.
114:    // int main(void)
115:    // -----
116:
117:    // Must be a multiple of 16
118:    .equ MAIN_STACK_BYTECOUNT, 32
119:
120:    // Local variable stack offsets:
121:    .equ LGCD, 8
122:    .equ L2, 16
123:    .equ L1, 24
124:
125:    .global main
126:

```



euclidopt.s (Page 3 of 3)

```
127: main:
128:
129:     // Prolog
130:     sub    sp, sp, MAIN_STACK_BYTECOUNT
131:     str    x30, [sp]
132:
133:     // long l1
134:     // long l2
135:     // long lGcd
136:
137:     // printf("Enter an integer: ")
138:     adr    x0, promptStr
139:     bl     printf
140:
141:     // scanf("%ld", &l1)
142:     adr    x0, scanfFormatStr
143:     add    x1, sp, L1
144:     bl     scanf
145:
146:     // printf("Enter an integer: ")
147:     adr    x0, promptStr
148:     bl     printf
149:
150:     // scanf("%ld", &l2)
151:     adr    x0, scanfFormatStr
152:     add    x1, sp, L2
153:     bl     scanf
154:
155:     // lGcd = gcd(l1, l2)
156:     ldr    x0, [sp, L1]
157:     ldr    x1, [sp, L2]
158:     bl     gcd
159:     str    x0, [sp, LGCD]
160:
161:     // printf("The gcd is %ld\n", lGcd)
162:     adr    x0, printfFormatStr
163:     ldr    x1, [sp, LGCD]
164:     bl     printf
165:
166:     // Epilog and return 0
167:     mov    w0, 0
168:     ldr    x30, [sp]
169:     add    sp, sp, MAIN_STACK_BYTECOUNT
170:     ret
171:
172:     .size main, (. - main)
```





## Precept Activity Instructions:

- `cd` into whatever directory you are using for precept exercises.
- `cp /u/cos217/PreceptDemos/rect.* .`
- Modify `rect.c` to use local variables instead of global variables. What changes do you need to make?
- Modify `rect.s` accordingly. What changes do you need to make?

12

Precept Activity solution code for rect.c with local variables: [Please see the Schedule page: Week 11, Wed/Thu precept handout rect.c.](#)

What changes did you make to modify rect.c:

Easy. Just move the three variables to the beginning of the main function.  
Need to remove the static keyword.

Precept Activity solution code for rect.s with local variables: [Please see the Schedule page: Week 11, Wed/Thu precept handout rect.s.](#)

What changes did you make to modify rect.s:

- 1) Delete labels (iLength, iWidth, and iPerim) in the bss section.
- 2) Determine the size of the stack frame: `MAIN_STACK_BYTECOUNT = 32`.  
8 bytes for X30, 4 for each local variable.  
total bytes needed:  $8+3*4 = 20$ .  
`MAIN_STACK_BYTECOUNT` must be a multiple of 16.
- 3) Define and determine local variable stack offsets. Use `.equ` to name offsets.
- 4) Use register immediate offset to access local variables on stack
- 5) Use "`add Xn, sp, offset`" to get an address of a local variable for the `scanf` function call.