P1

```
 1: /*--------------------------------------------------------------------*/
 2: /* absval.c                                                           */
 3: /* Author: Bob Dondero                                                */
 4: /*--------------------------------------------------------------------*/
 5:
 6: #include <stdio.h>
 7: #include <stdlib.h>
 8:
 9: /*--------------------------------------------------------------------*/
10:
11: static int iInput;   /* Bad style. */
12: static int iAbsVal;  /* Bad style. */
13:
14: /*--------------------------------------------------------------------*/
15:
16: /* Read an integer from stdin, and write its absolute value
17:    to stdout.  Return 0. */
18:
19: int main(void)
20: {
21:    printf("Enter an integer:  ");
22:    scanf("%d", &iInput);  /* Should validate. */
23:
24:    iAbsVal = abs(iInput);
25:
26:    printf("The integer's absolute value is %d.\n", iAbsVal);
27:
28:    return 0;
29: }
```

```
1: //-----------------------------------------------------------------------
2: // absval.s
3: // Author: Bob Dondero and William Ughetta
4: //-----------------------------------------------------------------------
5:
6:         .section .rodata
7:
8: promptStr:
9:         .string "Enter an integer:  "
10:
11: scanfFormatStr:
12:         .string "%d"
13:
14: printfFormatStr:
15:         .string "The integer's absolute value is %d.\n"
16:
17: //-----------------------------------------------------------------------
18:
19:         .section .data
20:
21: //-----------------------------------------------------------------------
22:
23:         .section .bss
24:
25: iInput:
26:         .skip   4
27:
28: iAbsVal:
29:         .skip   4
30:
31: //-----------------------------------------------------------------------
32:
33:         .section .text
34:
35:         //-------------------------------------------------------------
36:         // Read an integer from stdin, and write its absolute value
37:         // to stdout.  Return 0.
38:         // int main(void)
39:         //-------------------------------------------------------------
40:
41:         // Must be a multiple of 16
42:         .equ    MAIN_STACK_BYTECOUNT, 16
43:
44:         .global main
45:
46: main:
47:
48:         // Prolog
49:         sub     sp, sp, MAIN_STACK_BYTECOUNT
50:         str     x30, [sp]
51:
52:         // printf("Enter an integer:  ")
53:         adr     x0, promptStr
54:         bl      printf
55:
56:         // scanf("%d", &iInput)
57:         adr     x0, scanfFormatStr
58:         adr     x1, iInput
59:         bl      scanf
60:
61:         // iAbsVal = abs(iInput)
62:         adr     x0, iInput
63:         ldr     w0, [x0]
```

```
64:         bl      abs
65:         adr     x1, iAbsVal
66:         str     w0, [x1]
67:
68:         // printf("The integer's absolute value is %d.\n", iAbsVal)
69:         adr     x0, printfFormatStr
70:         adr     x1, iAbsVal
71:         ldr     w1, [x1]
72:         bl      printf
73:
74:         // Epilog and return 0
75:         mov     w0, 0
76:         ldr     x30, [sp]
77:         add     sp, sp, MAIN_STACK_BYTECOUNT
78:         ret
79:
80:         .size   main, (. - main)
```

uppercase.c (Page 1 of 1)

```
 1: /*---------------------------------------------------------------------*/
 2: /* uppercase.c                                                         */
 3: /* Author: Bob Dondero                                                 */
 4: /*---------------------------------------------------------------------*/
 5:
 6: #include <stdio.h>
 7: #include <ctype.h>
 8:
 9: /*---------------------------------------------------------------------*/
10:
11: static char cChar;   /* Bad style. */
12:
13: /*---------------------------------------------------------------------*/
14:
15: /* Read a character from stdin, and write its uppercase equivalent
16:    to stdout.  Return 0. */
17:
18: int main(void)
19: {
20:    cChar = (char)getchar();       /* Should check for error. */
21:
22:    cChar = (char)toupper((int)cChar);
23:
24:    putchar((int)cChar);
25:    putchar('\n');
26:
27:    return 0;
28: }
```

uppercase.s (Page 1 of 2)

```
 1: //-----------------------------------------------------------------------
 2: // uppercase.s
 3: // Author: Bob Dondero and William Ughetta
 4: //-----------------------------------------------------------------------
 5:
 6:         .section .rodata
 7:
 8: //-----------------------------------------------------------------------
 9:
10:         .section .data
11:
12: //-----------------------------------------------------------------------
13:
14:         .section .bss
15:
16: cChar:
17:         .skip   1
18:
19: //-----------------------------------------------------------------------
20:
21:         .section .text
22:
23:         //-------------------------------------------------------------
24:         // Read a letter from stdin, and write its uppercase
25:         // equivalent to stdout.  Return 0.
26:         // int main(void)
27:         //-------------------------------------------------------------
28:
29:         // Must be a multiple of 16
30:         .equ    MAIN_STACK_BYTECOUNT, 16
31:
32:         .global main
33:
34: main:
35:
36:         // Prolog
37:         sub     sp, sp, MAIN_STACK_BYTECOUNT
38:         str     x30, [sp]
39:
40:         // cChar = (char)getchar()
41:         bl      getchar
42:         adr     x1, cChar
43:         strb    w0, [x1]
44:
45:         // cChar = (char)toupper((int)cChar)
46:         adr     x1, cChar
47:         ldrb    w0, [x1]
48:         bl      toupper
49:         adr     x1, cChar
50:         strb    w0, [x1]
51:
52:         // putchar((int)cChar)
53:         adr     x1, cChar
54:         ldrb    w0, [x1]
55:         bl      putchar
56:
57:         // putchar('\n')
58:         mov     w0, '\n'
59:         bl      putchar
60:
61:         // Epilog and return 0
62:         mov     w0, 0
63:         ldr     x30, [sp]
```

```
64:         add     sp, sp, MAIN_STACK_BYTECOUNT
65:         ret
66:
67:         .size   main, (. - main)
```

rect.c (Page 1 of 1)

```
 1: /*------------------------------------------------------------------------*/
 2: /* rect.c                                                                 */
 3: /* Author: Bob Dondero                                                    */
 4: /*------------------------------------------------------------------------*/
 5:
 6: #include <stdio.h>
 7:
 8: /*------------------------------------------------------------------------*/
 9:
10: static int iLength;  /* Bad style. */
11: static int iWidth;   /* Bad style. */
12: static int iPerim;   /* Bad style. */
13:
14: /*------------------------------------------------------------------------*/
15:
16: /* Read a rectangle's length and width from stdin, and write its
17:    perimeter to stdout.  Return 0. */
18:
19: int main(void)
20: {
21:    printf("Rectangle length:  ");
22:    scanf("%d", &iLength);   /* Should validate. */
23:
24:    printf("Rectangle width:  ");
25:    scanf("%d", &iWidth);    /* Should validate. */
26:
27:    iPerim = 2 * (iLength + iWidth);
28:
29:    printf("The rectangle's perimeter is %d.\n", iPerim);
30:
31:    return 0;
32: }
```

```
1: //-----------------------------------------------------------------------
2: // rect.s
3: // Author: Bob Dondero and William Ughetta
4: //-----------------------------------------------------------------------
5:
6:         .section .rodata
7:
8: lengthPromptStr:
9:         .string "Rectangle length:  "
10:
11: widthPromptStr:
12:         .string "Rectangle width:  "
13:
14: scanfFormatStr:
15:         .string "%d"
16:
17: printfFormatStr:
18:         .string "The rectangle's perimeter is %d.\n"
19:
20: //-----------------------------------------------------------------------
21:
22:         .section .data
23:
24: //-----------------------------------------------------------------------
25:
26:         .section .bss
27:
28: iLength:
29:         .skip   4
30:
31: iWidth:
32:         .skip   4
33:
34: iPerim:
35:         .skip   4
36:
37: //-----------------------------------------------------------------------
38:
39:         .section .text
40:
41:         //-------------------------------------------------------------
42:         // Read a rectangles length and width from stdin, and write
43:         // its perimeter to stdout.  Return 0.
44:         // int main(void)
45:         //-------------------------------------------------------------
46:
47:         // Must be a multiple of 16
48:         .equ    MAIN_STACK_BYTECOUNT, 16
49:
50:         .global main
51:
52: main:
53:
54:         // Prolog
55:         sub     sp, sp, MAIN_STACK_BYTECOUNT
56:         str     x30, [sp]
57:
58:         // printf("Rectangle length:  ")
59:         adr     x0, lengthPromptStr
60:         bl      printf
61:
62:         // scanf("%d", &iLength)
63:         adr     x0, scanfFormatStr
```

```
64:         adr     x1, iLength
65:         bl      scanf
66:
67:         // printf("Rectangle width:  ")
68:         adr     x0, widthPromptStr
69:         bl      printf
70:
71:         // scanf("%d", &iWidth)
72:         adr     x0, scanfFormatStr
73:         adr     x1, iWidth
74:         bl      scanf
75:
76:         // iPerimeter = 2 * (iLength + iWidth)
77:         adr     x0, iLength
78:         ldr     w0, [x0]
79:         adr     x1, iWidth
80:         ldr     w1, [x1]
81:         add     w0, w0, w1
82:         lsl     w0, w0, 1
83:            // Alternatives to lsl (either of these two):
84:            //    add w0, w0, w0
85:            //    mov w1, 2; mul w0, w0, w1
86:         adr     x1, iPerim
87:         str     w0, [x1]
88:
89:         // printf("The rectangle's perimeter is %d.\n", iPerim)
90:         adr     x0, printfFormatStr
91:         adr     x1, iPerim
92:         ldr     w1, [x1]
93:         bl      printf
94:
95:         // Epilog and return 0
96:         mov     w0, 0
97:         ldr     x30, [sp]
98:         add     sp, sp, MAIN_STACK_BYTECOUNT
99:         ret
100:
101:        .size   main, (. - main)
```

P10

```
 1: /*--------------------------------------------------------------------*/
 2: /* power.c                                                            */
 3: /* Author: Bob Dondero                                                */
 4: /*--------------------------------------------------------------------*/
 5:
 6: #include <stdio.h>
 7:
 8: /*--------------------------------------------------------------------*/
 9:
10: static long lBase;       /* Bad style. */
11: static long lExp;        /* Bad style. */
12: static long lPower = 1;  /* Bad style. */
13: static long lIndex;      /* Bad style. */
14:
15: /*--------------------------------------------------------------------*/
16:
17: /* Read a non-negative base and exponent from stdin.  Write base
18:    raised to the exponent power to stdout.  Return 0. */
19:
20: int main(void)
21: {
22:    printf("Enter the base:  ");
23:    scanf("%ld", &lBase);   /* Should validate. */
24:
25:    printf("Enter the exponent:  ");
26:    scanf("%ld", &lExp);    /* Should validate. */
27:
28:    for (lIndex = 1; lIndex <= lExp; lIndex++)
29:       lPower *= lBase;
30:
31:    printf("%ld to the %ld power is %ld.\n", lBase, lExp, lPower);
32:
33:    return 0;
34: }
```

Princeton University
COS 217: Introduction to Programming Systems
"Flattened" C Programs

C Code	"Flattened" C Code

`if (expression)`
`{`
`  statement1;`
`  ...`
`  statementN;`
`}`
	`  if (! expression) goto endif1;`
	`  statement1;`
	`  ...`
	`  statementN;`
	`endif1:`

`if (expression)`
`{`
`  statementT1;`
`  ...`
`  statementTN;`
`}`
`else`
`{`
`  statementF1;`
`  ...`
`  statementFN;`
`}`
	`  if (! expression) goto else1;`
	`  statementT1;`
	`  ...`
	`  statementTN;`
	`  goto endif1;`
	`else1:`
	`  statementF1;`
	`  ...`
	`  statementFN;`
	`endif1:`

`while (expression)`
`{`
`  statement1;`
`  ...`
`  statementN;`
`}`
	`loop1:`
	`  if (! expression) goto endloop1;`
	`  statement1;`
	`  ...`
	`  statementN;`
	`  goto loop1;`
	`endloop1:`

`for (expression1; expression2; expression3)`
`{`
`  statement1;`
`  ...`
`  statementN;`
`}`
	`  expression1;`
	`loop1:`
	`  if (! expression2) goto endloop1;`
	`  statement1;`
	`  ...`
	`  statementN;`
	`  expression3;`
	`  goto loop1;`
	`endloop1:`

A loop pattern that is more efficient is described in Section 3.2 of *ARM 64-bit Assembly* Language by Pyeatt with Ughetta.

powerflat.c (Page 1 of 1)

```
 1: /*--------------------------------------------------------------------*/
 2: /* powerflat.c                                                        */
 3: /* Author: Bob Dondero                                                */
 4: /*--------------------------------------------------------------------*/
 5:
 6: #include <stdio.h>
 7:
 8: /*--------------------------------------------------------------------*/
 9:
10: static long lBase;       /* Bad style. */
11: static long lExp;        /* Bad style. */
12: static long lPower = 1;  /* Bad style. */
13: static long lIndex;      /* Bad style. */
14:
15: /*--------------------------------------------------------------------*/
16:
17: /* Read a non-negative base and exponent from stdin.  Write base
18:    raised to the exponent power to stdout.  Return 0. */
19:
20: int main(void)
21: {
22:    printf("Enter the base:  ");
23:    scanf("%ld", &lBase);
24:
25:    printf("Enter the exponent:  ");
26:    scanf("%ld", &lExp);
27:
28:    lIndex = 1;
29: powerLoop:
30:    if (lIndex > lExp) goto powerLoopEnd;
31:    lPower *= lBase;
32:    lIndex++;
33:    goto powerLoop;
34: powerLoopEnd:
35:
36:    printf("%ld to the %ld power is %ld.\n", lBase, lExp, lPower);
37:
38:    return 0;
39: }
```

```
1: //-----------------------------------------------------------------------
2: // power.s
3: // Author: Bob Dondero and William Ughetta
4: //-----------------------------------------------------------------------
5:
6:         .section .rodata
7:
8: basePromptStr:
9:         .string "Enter the base:  "
10:
11: expPromptStr:
12:         .string "Enter the exponent:  "
13:
14: scanfFormatStr:
15:         .string "%ld"
16:
17: printfFormatStr:
18:         .string "%ld to the %ld power is %ld.\n"
19:
20: //-----------------------------------------------------------------------
21:
22:         .section .data
23:
24: lPower:
25:         .quad   1
26:
27: //-----------------------------------------------------------------------
28:
29:         .section .bss
30:
31: lBase:
32:         .skip   8
33:
34: lExp:
35:         .skip   8
36:
37: lIndex:
38:         .skip   8
39:
40: //-----------------------------------------------------------------------
41:
42:         .section .text
43:
44:         //-------------------------------------------------------------
45:         // Read a non-negative base and exponent from stdin.  Write
46:         // base raised to the exponent power to stdout.  Return 0.
47:         // int main(void)
48:         //-------------------------------------------------------------
49:
50:         // Must be a multiple of 16
51:         .equ    MAIN_STACK_BYTECOUNT, 16
52:
53:         .global main
54:
55: main:
56:
57:         // Prolog
58:         sub     sp, sp, MAIN_STACK_BYTECOUNT
59:         str     x30, [sp]
60:
61:         // printf("Enter the base:  ")
62:         adr     x0, basePromptStr
63:         bl      printf
```

```
64:
65:         // scanf("%d", &lBase)
66:         adr     x0, scanfFormatStr
67:         adr     x1, lBase
68:         bl      scanf
69:
70:         // printf("Enter the exponent:  ")
71:         adr     x0, expPromptStr
72:         bl      printf
73:
74:         // scanf("%d", &lExp)
75:         adr     x0, scanfFormatStr
76:         adr     x1, lExp
77:         bl      scanf
78:
79:         // lIndex = 1
80:         mov     x0, 1
81:         adr     x1, lIndex
82:         str     x0, [x1]
83:
84: powerLoop:
85:
86:         // if (lIndex > lExp) goto powerLoopEnd
87:         adr     x0, lIndex
88:         ldr     x0, [x0]
89:         adr     x1, lExp
90:         ldr     x1, [x1]
91:         cmp     x0, x1
92:         bgt     powerLoopEnd
93:
94:         // lPower *= lBase
95:         adr     x0, lPower
96:         ldr     x1, [x0]
97:         adr     x2, lBase
98:         ldr     x2, [x2]
99:         mul     x1, x1, x2
100:        str     x1, [x0]
101:
102:        // lIndex++
103:        adr     x0, lIndex
104:        ldr     x1, [x0]
105:        add     x1, x1, 1
106:        str     x1, [x0]
107:
108:        // goto powerLoop
109:        b       powerLoop
110:
111: powerLoopEnd:
112:
113:        // printf("%ld to the %ld power is %ld.\n", lBase, lExp, lPower)
114:        adr     x0, printfFormatStr
115:        adr     x1, lBase
116:        ldr     x1, [x1]
117:        adr     x2, lExp
118:        ldr     x2, [x2]
119:        adr     x3, lPower
120:        ldr     x3, [x3]
121:        bl      printf
122:
123:        // Epilog and return 0
124:        mov     w0, 0
125:        ldr     x30, [sp]
126:        add     sp, sp, MAIN_STACK_BYTECOUNT
```

```
127:        ret
128:
129:        .size   main, (. - main)
```