```
;; Set R0 to 10*R1
      .ORIG x3000
mul10 ADD   R0,R1,R1
      ADD   R0,R0,R0
      ADD   R0,R0,R1
      ADD   R0,R0,R0
      HALT
```

```
;; Set R3 to R1 XOR R2
      .ORIG x3000
xor   NOT   R1,R1
      AND   R3,R1,R2
      NOT   R1,R1
      NOT   R2,R2
      AND   R4,R1,R2
      NOT   R2,R2
      NOT   R3,R3
      NOT   R4,R4
      AND   R3,R3,R4
      NOT   R3,R3
      HALT
```

```
;; Set R0 to the number of bits “on” in R1
      .ORIG x3000
pop   AND   R0,R0,#0
      ADD   R1,R1,#0    ;; Do the high bit first
      BRzp  skipf
      ADD   R0,R0,#1
skipf AND   R2,R2,#0
      ADD   R2,R2,#15   ;; R2 <- 15
loop  ADD   R1,R1,R1
      BRzp  skip
      ADD   R0,R0,#1
skip  ADD   R2,R2,#-1
      BRp   loop
      HALT
      .END
```

```
;; Sets r to
;; <0, if a<b
;; =0, if a==b
;; >0, if a>b
      .ORIG x3000
      LD    R1,a
      LD    R2,b
      NOT   R2,R2
      ADD   R2,R2,#1
      ADD   R2,R1,R2      ;; R2 <- a-b
      ST    R2,r
      HALT
a     .FILL #109
b     .FILL #209
r     .BLKW 1
      .END
```

```
;; Really sets r to
;; <0, if a<b
;;  0, if a==b
;; >0, if a>b


;;          |  a<0  |  a>=0 |
;;  ------ +-------+-------+
;;   b<0   |  cmp  |   A   |
;;  ------ +-------+-------+
;;   b>=0  |   A   |  cmp  |
;;  ------ +-------+-------+

       .ORIG    x3000
cint    LD      R1,a
        BRn     aNeg

;; a>=0, if here
        LD      R2,b
        BRn     retA
        BR      cmp

;; a<0, if here
aNeg    LD      R2,b
        BRn     cmp
;;      BR      retA

;; a and b have different signs
retA    ST      R1,r
        BR     leave

;; a and b have sign sign
cmp     NOT    R2,R2
        ADD    R2,R2,#1
        ADD    R2,R1,R2
        ST     R2,r

leave   HALT

a      .FILL   #-20000
b      .FILL   #20000
r      .BLKW   1
       .END
```

```
;; Really sets r to
;; <0, if a<b
;;  0, if a==b
;; >0, if a>b


;;          |  a<0  |  a>=0 |
;;  ------ +-------+-------+
;;   b<0   |  cmp  |   A   |
;;  ------ +-------+-------+
;;   b>=0  |   A   |  cmp  |
;;  ------ +-------+-------+

       .ORIG    x3000
cint    LD      R3,b15
        LD      R1,a
        LD      R2,b
        AND     R3,R3,R1
        ADD     R3,R3,R2
        BRzp    cmp

;; a and b have different signs
retA    ST      R1,r
        BR     leave

;; a and b have same sign
cmp     NOT    R2,R2
        ADD    R2,R2,#1
        ADD    R2,R1,R2
        ST     R2,r

leave   HALT

b15    .FILL   x8000
a      .FILL   #-20000
b      .FILL   #20000
r      .BLKW   1
       .END
```

```
;; Counts the number of times a character occurs in a string
;; Character, string, and result are all stored nearby
        .ORIG    x3000
nmChr   AND      R0,R0,#0
        LEA      R1,FILE
        LD       R2,LOOK4
        NOT      R2,R2
        ADD      R2,R2,#1
ALOOP   LDR      R3,R1,#0
        BRz      STOPIT
        ADD      R3,R3,R2
        BRnp     NOCOUNT
        ADD      R0,R0,#1
NOCOUNT ADD      R1,R1,#1
      BR       ALOOP
STOPIT  ST       R0,COUNT
        HALT
LOOK4   .FILL    x73
FILE    .STRINGZ "This is such fun!"
COUNT   .BLKW    1
        .END
```

```
;; Counts the number of times a character occurs in a string
;; Character -- stored at x4000
;; String    -- stored at x5000
;; Result    -- stored at x6000
        .ORIG    x3000
nmChr   AND      R0,R0,#0
        LD       R1,AFILE
        LDI      R2,ALOOK4
        NOT      R2,R2
        ADD      R2,R2,#1
ALOOP   LDR      R3,R1,#0
        BRz      STOPIT
        ADD      R3,R3,R2
        BRnp     NOCOUNT
        ADD      R0,R0,#1
NOCOUNT ADD      R1,R1,#1
        BR       ALOOP
STOPIT  STI      R0,ACOUNT
        HALT
ALOOK4  .FILL    x4000
AFILE   .FILL    x5000
ACOUNT  .FILL    x6000
        .END
```

```
;; This must be in a separate file
        .ORIG    x5000
FILE    .STRINGZ "This is such fun!"
        .END
```

```
;; AND's a vector of words
        .ORIG    x3000
andV    AND      R0,R0,#0
        ADD      R0,R0,#-1
        LD       R1,SIZE
        LEA      R2,VECT
ALOOP   ADD      R1,R1,#-1
        BRn      STOPIT
        LDR      R3,R2,#0
        AND      R0,R0,R3
        ADD      R2,R2,#1
        BR       ALOOP
STOPIT  ST       R0,RESULT
        HALT
SIZE    .FILL    5
VECT    .FILL    xBEEF
        .FILL    x89AB
        .FILL    xFFFF
        .FILL    x89AB
        .FILL    x2008
RESULT  .BLKW    1
        .END
```

```
;; Reverse a string
        .ORIG    x3000
rev     LEA      R0,FILE      ;; R0 is beginning of string
        ADD      R1,R0,#-1    ;; R1 will point to end of string
LOOP1   LDR      R3,R1,#1
        BRz      DONE1
        ADD      R1,R1,#1
        BR       LOOP1

DONE1   NOT      R2,R0
        ADD      R2,R2,R1

;; R0 == address of first character of string
;; R1 == address of last character of string
;; R2 == size of string - 2  (Think about it....)
LOOP2   ADD      R2,R2,#0
        BRn      DONE2
        LDR      R3,R0,#0     ;; Swap
        LDR      R4,R1,#0
        STR      R4,R0,#0
        STR      R3,R1,#0
        ADD      R0,R0,#1     ;; move pointers
        ADD      R1,R1,#-1
        ADD      R2,R2,#-2
        BR       LOOP2

DONE2   HALT

FILE    .STRINGZ "CSCI is such fun!"
        .END
```

```
;; Assume ALL numbers are unsigned!!!

;; Assume R2 contains a number to be divided by 10
```
;; Put the Quotient in R0 and the remainder in R1
```
        .ORIG    x3000
div10   ADD  R0,R2,#0           ;; R0 <- R2
        AND      R1,R1,#0           ;; R1 <-  0
        AND      R3,R3,#0
        ADD      R3,R3,#4           ;; R3 <-  4

;; left shift R0 four bits,
;; put bits shifted out into R1
SHFT4   ADD      R1,R1,R1
        ADD      R0,R0,#0
        BRzp     noSetA
        ADD      R1,R1,#1
noSetA  ADD      R0,R0,R0
        ADD      R3,R3,#-1
        BRp      SHFT4


;; At the beinning of this loop
;;   R2 = abcd efgh ijkl mnop
;;   R1 = 0000 0000 0000 abcd
;;   R0 = efgh ijkl mnop 0000
;; As we go through the loop, we test R1
;;   If R1>10,
;;      we need to put a 1 in the quotient
;;        being formed at the end of R0,
;;      and we need to subract 10 from R1
;;   Then shift R1-R0 as a unit, placing
;;     msb of R0 into R1


;; Go through the divide loop 12 times
        ADD      R3,R3,#12          ;; R3 <- 12
;;   Testing partial remainder, to see if it is greater than 10
DIVLP   ADD      R4,R1,#-10         ;; test if bits in R1 > 10
        BRn      lt10
        ADD      R0,R0,#1           ;; if so, add 1 to R0
        ADD      R1,R4,#0           ;; and set R1 <- R1-10
lt10    ADD      R1,R1,R1           ;; shift R1 over 1 position
        ADD      R0,R0,#0           ;; move msb or R0 to lsb of R1
        Brzp     noSetB
        ADD      R1,R1,#1
noSetB  ADD      R0,R0,R0           ;; shift R0 over 1 position
        ADD      R3,R3,#-1
        BRp      DIVLP

;; Need to make one more adjustment ...
        ADD      R4,R1,#-10
        BRn      DONE
        ADD      R0,R0,#1
        ADD      R1,R4,#0

DONE    HALT

        .END
```