9.2

a. 256. Since the trap vector is 8-bit wide.

b. (1) Using a RET instruction, the PC can be loaded with any address. However, using a BR instruction, the PC can only be loaded with addresses within a limited range. (2) Program control can certainly be brought back to the caller program no matter how many times a TRAP instruction is called because R7 could contain any address. If a BR instruction is used, the PC could only be loaded with a certain address no matter where a TRAP instruction is called, which means the caller program cannot continues from where it calls the TRAP instruction.

c. One.

9.3

a. An external mechanism (outside the CPU) may be needed to restart the clock and set MCR[15] to 1.

b. STI R0, MCR

c. LD R1, SaveR1

d. The caller program that calls the HALT instruction.

9.8

If the value in A is a prime, the location RESULT contains 1. Otherwise, the location RESULT contains 0.

9.9

a. The subroutine stores the result in R0 and is called NOBUSY.

```
NOBUSY      ST      R1, SaveR1
            ST      R2, SaveR2
            AND     R0, R0, #0      ; clear R0
            LDI     R1, VECT        ; load the bit pattern
            LD      R2, MASK        ; load the mask
            AND     R1, R1, R2
            BRnp    END
            ADD     R0, R0, #1
END         LD      R1, SaveR1
            LD      R2, SaveR2
            RET
SaveR1      .BLKW   1
SaveR2      .BLKW   1
VECT        .FILL   x4001
MASK        .FILL   x00FF
```

9.13

Because the linkage back to JSR A is destroyed when the subroutine B is executed.

9.15

a. TRAP x72

b. Yes. Each TRAP instruction in this routine saves the value of R7, so the program control can successfully come back to this routine and finally return to the caller program without anything destroyed, except the value in R0.

9.17

```
(a) LD     R3, NEGENTER
(b) STR    R0, R1, #0
(c) ADD    R1, R1, #1
(d) STR    R2, R1, #0
```

9.18

```
(a) ADD    R1, R1, #1
(b) TRAP   x25
(c) ADD    R0, R0, #5
(d) BRzp   K
```

10.3

```
(a) PUSH   R1
```

(*b*) POP R0

(*c*) PUSH R3

(*d*) POP R7

10.4

The value is stored in R0. If underflow occurs, R1 will contain 1. Otherwise, R1 contains 0. Overflow error checking is unnecessary because the function is not inserting a value into the stack and so overflow will never happen. I assume that R6 is pointing to the first element already and that the base of the stack is x4000. The function is as follow:

```
PEEK        AND    R1, R1, 0      ; clear R1
            LEA    R0, BASE
            NOT    R0, R0
            ADD    R0, R0, #1
            ADD    R0, R0, R6     ; compare two addresses
            BRz    UNDERFLOW
            LDR    R0, R6, #0     ; load the top element
            RET
UNDERFLOW   ADD    R1, R1, #1     ; underflow
            RET
BASE        .FILL  x4000
```

10.11

x01F1 contains x6200. x01F2 contains x6300. They both belong to the Interrupt Vector Table.

10.23

This program displays the input sentence in reverse order. (The input sentence should end with an enter.)