How to achieve looping in assembly language (Hints: assembly language has no while or for loop built in)?

Specifically, what are the loop conditions (initialize value, end value, increment on each step…)

BUN to go to SNA part, if value is negative, skip next line, if positive, BUN to 2 and runs the body till it's negative. (negative since I used SNA)

Basic step:

Take a two digit input (Octal number)
Convert the octal number to decimal number
If it is a prime number
    display the number
Else
    Display 0

Looping **TABLE 6-5**

```
1       BUN   6
2
3
4
5
6       SNA
7       BUN   2
8
```

Subroutine **TABLE 6-16**

```
                ORG 1
1               LDA X
2               BSA SH1
3               STA X
4               HLT
5
6       X,      HEX 1234
7
8       SH1     HEX 0
9
10      1
11
12
```

Taking input and storing: TABLE 6-20

```
        ORG   10
FST, SKI              // look for input
        BUN   FST     // no input
```

```
        CLA             // clear AC
        INP             // input to AC
        ADD    ECH
        OUT             // output char
        HLT             // quit
ECH, DEC 2              // Stores decimal number 2
        END             // End of the program
```

Converting octal number to decimal number

Since the input is just two digits.
(input / 10) * 32 + (input % 10)

Division (adding a to a for n times until it's greater than equal to b)

```
MOV AX, 100    ; Load the numerator into the AX register
MOV BX, 5      ; Load the denominator into the BX register
DIV BX         ; Divide AX by BX; quotient is stored in AX, remainder in DX
```

Multiplying (adding a to a for b times) TABLE 6-14

```
mov ax, [a]
mov bx, [b]

; Multiply the values and store the result in result
imul bx           ; Multiply AX by BX (signed multiplication)
mov [result], ax  ; Store the result in the result variable
```

Adding TABLE 6-15

```
; Load the values of a and b into registers
mov ax, [a]
mov bx, [b]
; Add the values and store the result in result
add ax, bx
mov [result], ax  ; Store the result in the result variable
```

Mod

```
remainder = dividend;
while (remainder >= divisor) {
remainder -= divisor;
}
return remainder;

mov ax, [dividend]   ; Load the dividend value into a register
mov bl, [divisor]    ; Load the divisor value into a register
```

```
; Perform the division operation and get the remainder
xor dx, dx          ; Clear the DX register (which will be used as the remainder)
div bl              ; Divide the AX register (dividend) by the BL register (divisor)
mov ax, dx          ; Move the remainder (stored in DX) into AX for outputting
```

To check the primality

```
int n; /* input number */
int i; /* loop counter */
int c = 0; /* the number of factor a number has */
printf("Enter a number n:");
scanf("%d", &n);
for (i = 1; i <= n; i++) {
        if (n % i == 0) {
                c++;
        }
}
if (c == 2) {
        printf("%d", n);
}
else {
        printf("0");
}
```