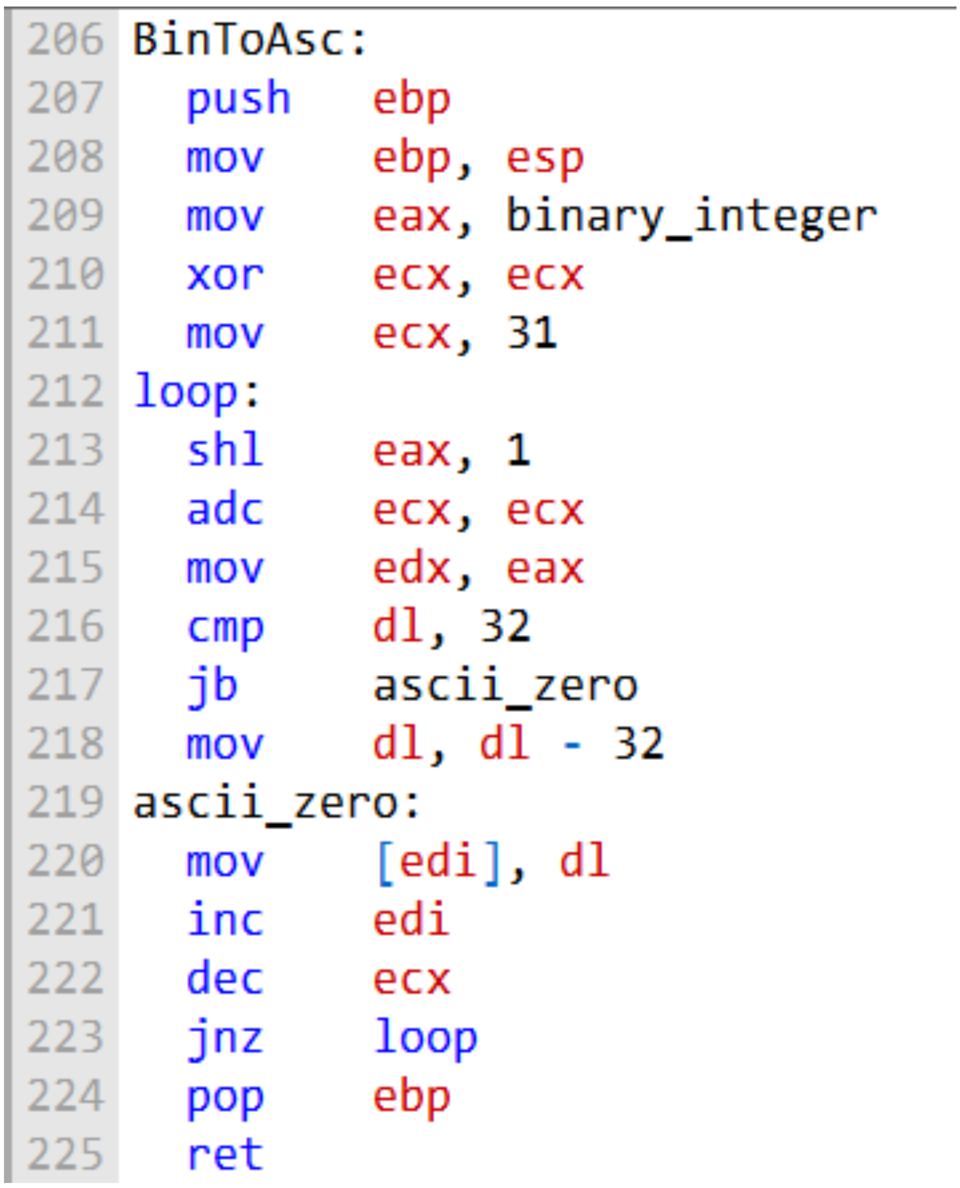
BinToAsc



```
BinToAsc:
  push   ebp
  mov    ebp, esp
  mov    eax, binary_integer
  xor    ecx, ecx
  mov    ecx, 31
loop:
  shl    eax, 1
  adc    ecx, ecx
  mov    edx, eax
  cmp    dl, 32
  jb     ascii_zero
  mov    dl, dl - 32
ascii_zero:
  mov    [edi], dl
  inc    edi
  dec    ecx
  jnz    loop
  pop    ebp
  ret
```

This procedure works by iterating over the bits in the binary integer, starting with the most significant bit.

For each bit, the procedure shifts the binary integer left by 1 bit and adds the carry flag to the counter register (ECX).

The carry flag is used to keep track of whether the previous iteration resulted in a carry-out.

If the binary integer is less than 32, then the least significant bit will be 0 and the carry flag will be 0. In this case, the procedure will move the ASCII character '0' (0x30) to the buffer at the address specified by the register EDI.

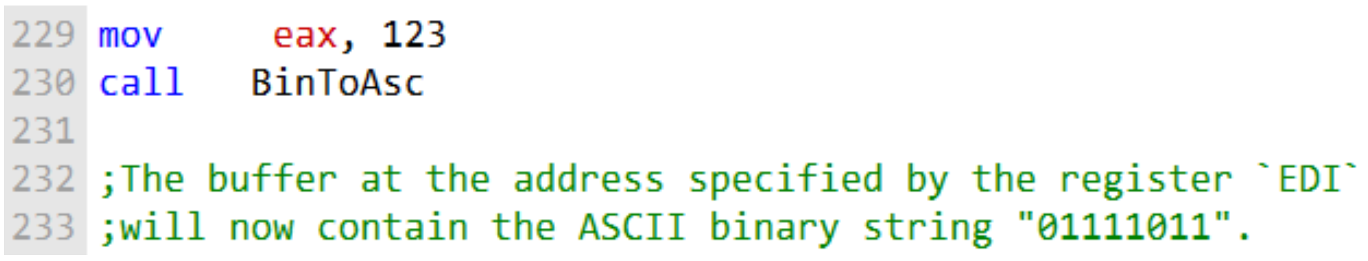
If the binary integer is greater than or equal to 32, then the least significant bit will be 1 and the carry flag will be 1.

In this case, the procedure will move the ASCII character '1' (0x31) to the buffer at the address specified by the register EDI.

After the procedure has finished iterating over the bits in the binary integer, the buffer at the address specified by the register EDI will contain the ASCII binary string representation of the binary integer.

Example 2 Usage:

The following code snippet shows how to use the BinToAsc procedure to convert the binary integer 123 (01111011) to an ASCII binary string:



```
mov     eax, 123
call   BinToAsc

;The buffer at the address specified by the register `EDI`
;will now contain the ASCII binary string "01111011".
```

The BinToAsc procedure is a simple and efficient way to convert a binary integer to an ASCII binary string. It is useful for displaying binary data on the console or in a file.