```
 1:  TITLE   8-bit multiplication using shifts   SHL_MLT.ASM
 2:  COMMENT |
 3:          Objective: To multiply two 8-bit unsigned numbers
 4:                     using SHL rather than MUL instruction.
 5:              Input: Requests two unsigned numbers from user.
 6:  |           Output: Prints the multiplication result.
 7:  .MODEL SMALL
 8:  .STACK 100H
 9:  .DATA
10:  input_prompt   DB  'Please input two short numbers: ',0
11:  out_msg1       DB  'The multiplication result is: ',0
12:  query_msg      DB  'Do you want to quit (Y/N): ',0
13:
14:  .CODE
15:  INCLUDE io.mac
16:  main    PROC
17:          .STARTUP
18:  read_input:
19:          PutStr  input_prompt ; request two numbers
20:          GetInt  AX           ; read the first number
21:          nwln
22:          GetInt  BX           ; read the second number
23:          nwln
```

Logical: 1

```
24:          call    mult8        ; mult8 uses SHL instruction
25:          PutStr  out_msg1
26:          PutInt  AX           ; mult8 leaves result in AX
27:          nwln
28:          PutStr  query_msg    ; query user whether to terminate
29:          GetCh   AL           ; read response
30:          nwln
31:          cmp     AL,'Y'       ; if response is not 'Y'
32:          jne     read_input   ; repeat the loop
33:  done:                        ; otherwise, terminate program
34:          .EXIT
35:  main    ENDP
36:
37:  ;-------------------------------------------------------
38:  ; mult8 multiplies two 8-bit unsigned numbers passed on to
39:  ; it in registers AL and BL. The 16-bit result is returned
40:  ; in AX. This procedure uses only SHL instruction to do the
41:  ; multiplication. All registers, except AX, are preserved.
42:  ;-------------------------------------------------------
43:  mult8  PROC
44:         push    CX           ; save registers
45:         push    DX
46:         push    SI
```

Logical: 2

```
47:         xor     DX,DX         ; DX := 0 (keeps mult. result)
48:         mov     CX,7          ; CX := # of shifts required
49:         mov     SI,AX         ; save original number in SI
50:  repeat1:       ; multiply loop - iterates 7 times
51:         rol     BL,1          ; test bits of number2 from left
52:         jnc     skip1         ; if 0, do nothing
53:         mov     AX,SI         ; else, AX := number1*bit weight
54:         shl     AX,CL
55:         add     DX,AX         ; update running total in DX
56:  skip1:
57:         loop    repeat1
58:         rol     BL,1          ; test the rightmost bit of AL
59:         jnc     skip2         ; if 0, do nothing
60:         add     DX,SI         ; else, add number1
61:  skip2:
62:         mov     AX,DX         ; move final result into AX
63:         pop     SI            ; restore registers
64:         pop     DX
65:         pop     CX
66:         ret
67:  mult8  ENDP
68:         END   main
```

Logical: 3

```
 1:  ;-------------------------------------------------------
 2:  ; mult8 multiplies two 8-bit unsigned numbers passed on to
 3:  ; it in registers AL and BL. The 16-bit result is returned
 4:  ; in AX. This procedure uses only SHL instruction to do the
 5:  ; multiplication. All registers, except AX, are preserved.
 6:  ; Demonstrates the use of bit instructions BSF and BTC.
 7:  ;-------------------------------------------------------
 8:  mult8  PROC
 9:         push    CX            ; save registers
10:         push    DX
11:         push    SI
12:         xor     DX,DX         ; DX := 0 (keeps mult. result)
13:         mov     SI,AX         ; save original number in SI
14:  repeat1:
15:         bsf     CX,BX         ; returns first 1 bit position in CX
16:         jz      skip1         ; if ZF=1, no 1 bit in BX - done
17:         mov     AX,SI         ; else, AX := number1*bit weight
18:         shl     AX,CL
19:         add     DX,AX         ; update running total in DX
20:         btc     BX,CX         ; complement the bit found by BSF
21:         jmp     repeat1
22:  skip1:
```

Logical: 4

```
22:  skip1:
23:          mov     AX,DX          ; move final result into AX
24:          pop     SI             ; restore registers
25:          pop     DX
26:          pop     CX
27:          ret
28:  mult8   ENDP
```

Logical: 5

```
 1:  TITLE   Octal-to-binary conversion using shifts   OCT_BIN.ASM
 2:  COMMENT |
 3:          Objective: To convert an 8-bit octal number to the
 4:                     binary equivalent using shift instruction.
 5:              Input: Requests an 8-bit octal number from user.
 6:             Output: Prints the decimal equivalent of the input
 7:  |                  octal number.
 8:  .MODEL SMALL
 9:  .STACK 100H
10:  .DATA
11:  octal_number   DB  4 DUP (?) ; to store octal number
12:  input_prompt   DB  'Please input an octal number: ',0
13:  out_msg1       DB  'The decimal value is: ',0
14:  query_msg      DB  'Do you want to quit (Y/N): ',0
15:
16:  .CODE
17:  INCLUDE io.mac
18:  main  PROC
19:        .STARTUP
20:  read_input:
21:        PutStr  input_prompt    ; request an octal number
22:        GetStr  octal_number,4  ; read input number
23:        nwln
```

Logical: 6

```
24:        mov     BX,OFFSET octal_number ; pass octal # pointer
25:        call    to_binary    ; returns binary value in AX
26:        PutStr  out_msg1
27:        PutInt  AX           ; display the result
28:        nwln
29:        PutStr  query_msg    ; query user whether to terminate
30:        GetCh   AL           ; read response
31:        nwln
32:        cmp     AL,'Y'       ; if response is not 'Y'
33:        jne     read_input   ; read another number
34:  done:                      ; otherwise, terminate program
35:        .EXIT
36:  main  ENDP
37:
38:  ;-----------------------------------------------------------
39:  ; to_binary receives a pointer to an octal number string in
40:  ; BX register and returns the binary equivalent in AL (AH is
41:  ; set to zero). Uses SHL for multiplication by 8. Preserves
42:  ; all registers, except AX.
43:  ;-----------------------------------------------------------
44:  to_binary     PROC
45:        push    BX           ; save registers
46:        push    CX
47:        push    DX
```

Logical: 7

```
48:        xor     AX,AX        ; result := 0
49:        mov     CX,3         ; max. number of octal digits
50:  repeat1:
51:        ; loop itarates a maximum of 3 times;
52:        ; but a NULL can terminates it early
53:        mov     DL,[BX]      ; read the octal digit
54:        cmp     DL,0         ; is it NULL?
55:        je      finished     ; if so, terminate loop
56:        and     DL,0FH       ; else, convert char. to numeric
57:        shl     AL,3         ; multiply by 8 and add to binary
58:        add     AL,DL
59:        inc     BX           ; move to next octal digit
60:        loop    repeat1      ; and repeat
61:  finished:
62:        pop     DX            ; restore registers
63:        pop     CX
64:        pop     BX
65:        ret
66:  to_binary     ENDP
67:        END     main
```

Logical: 8