

```

1:  TITLE   Binary equivalent of characters   BINCHAR.ASM
2:  COMMENT |
3:          Objective: To print the binary equivalent of
4:          ASCII character code.
5:          Input: Requests a character from keyboard.
6:          Output: Prints the ASCII code of the
7:          |
8:          input character in binary.
9:  .MODEL  SMALL
10: .STACK  100H
11: .DATA
12: char_prompt  DB  'Please input a character: ',0
13: out_msg1     DB  'The ASCII code of ''',0
14: out_msg2     DB  ''' in binary is ',0
15: query_msg    DB  'Do you want to quit (Y/N): ',0
16:
17: .CODE
18: INCLUDE io.mac
19: main  PROC
20:     .STARTUP
21:     read_char:
22:         PutStr  char_prompt  ; request a char. input
23:         GetCh   AL           ; read input character
24:         nwnln
25:         PutStr  out_msg1
26:         PutCh   AL

```

Overview: 1

```

26:         PutStr  out_msg2
27:         mov     AH,80H      ; mask byte = 80H
28:         mov     CX,8       ; loop count to print 8 bits
29:     print_bit:
30:         test   AL,AH      ; test does not modify AL
31:         jz     print_0    ; if tested bit is 0, print it
32:         PutCh  '1'       ; otherwise, print 1
33:         jmp    skip1
34:     print_0:
35:         PutCh  '0'       ; print 0
36:     skip1:
37:         shr    AH,1      ; right shift mask bit to test
38:         ; next bit of the ASCII code
39:         loop   print_bit
40:         nwnln
41:         PutStr  query_msg ; query user whether to terminate
42:         GetCh  AL       ; read response
43:         nwnln
44:         cmp    AL,'Y'    ; if response is not 'Y'
45:         jne    read_char ; read another character
46:     done:
47:         .EXIT
48:     main  ENDP
49:         END    main

```

Overview: 2

```

1: TITLE   Hex equivalent of characters   HEX1CHAR.ASM
2: COMMENT |
3:         Objective: To print the hex equivalent of
4:         ASCII character code.
5:         Input: Requests a character from keyboard.
6:         Output: Prints the ASCII code of the
7:         |
8:         input character in hex.
9: .MODEL  SMALL
10: .STACK 100H
11: .DATA
12: char_prompt  DB 'Please input a character: ',0
13: out_msg1     DB 'The ASCII code of ',0
14: out_msg2     DB ' ' in hex is ',0
15: query_msg    DB 'Do you want to quit (Y/N): ',0
16: .CODE
17: .486
18: INCLUDE io.mac
19: main  PROC
20:     .STARTUP
21: read_char:
22:     PutStr  char_prompt  ; request a char. input
23:     GetCh   AL           ; read input character
24:     nwlLn
25:     PutStr  out_msg1
26:     PutCh   AL
27:     PutStr  out_msg2

```

Overview: 3

```

28:     mov     AH,AL      ; save input character in AH
29:     shr     AL,4       ; move upper 4 bits to lower half
30:     mov     CX,2       ; loop count - 2 hex digits to print
31: print_digit:
32:     cmp     AL,9       ; if greater than 9
33:     jg     A_to_F     ; convert to A through F digits
34:     add     AL,'0'     ; otherwise, convert to 0 through 9
35:     jmp     skip
36: A_to_F:
37:     add     AL,'A'-10  ; subtract 10 and add 'A'
38:     ; to convert to A through F
39: skip:
40:     PutCh  AL         ; write the first hex digit
41:     mov     AL,AH     ; restore input character in AL
42:     and     AL,0FH    ; mask off the upper half byte
43:     loop   print_digit
44:     nwlLn
45:     PutStr query_msg  ; query user whether to terminate
46:     GetCh  AL         ; read response
47:     nwlLn
48:     cmp     AL,'Y'    ; if response is not 'Y'
49:     jne    read_char  ; read another character
50: done:
51:     .EXIT
52: main  ENDP
53:     END    main

```

Overview: 4

```

1: TITLE   Hex equivalent of characters   HEX2CHAR.ASM
2: COMMENT |
3:         Objective: To print the hex equivalent of
4:           ASCII character code. Demonstrates
5:           the use of xlat instruction.
6:         Input: Requests a character from keyboard.
7:         Output: Prints the ASCII code of the
8:         |           input character in hex.
9: .MODEL  SMALL
10: .STACK 100H
11: .DATA
12: char_prompt  DB  'Please input a character: ',0
13: out_msg1     DB  'The ASCII code of ''',0
14: out_msg2     DB  ''' in hex is ',0
15: query_msg    DB  'Do you want to quit (Y/N): ',0
16: ; translation table: 4-bit binary to hex
17: hex_table    DB  '0123456789ABCDEF'
18:
19: .CODE
20: .486
21: INCLUDE io.mac
22: main        PROC
23:             .STARTUP

```

Overview: 5

```

24: read_char:
25:     PutStr  char_prompt  ; request a char. input
26:     GetCh   AL           ; read input character
27:     nWln
28:     PutStr  out_msg1
29:     PutCh   AL
30:     PutStr  out_msg2
31:     mov     AH,AL        ; save input character in AH
32:     mov     BX,OFFSET hex_table ; BX := translation table
33:     shr     AL,4         ; move upper 4 bits to lower half
34:     xlatb                    ; replace AL with hex digit
35:     PutCh   AL           ; write the first hex digit
36:     mov     AL,AH        ; restore input character to AL
37:     and     AL,0FH       ; mask off upper 4 bits
38:     xlatb
39:     PutCh   AL           ; write the second hex digit
40:     nWln
41:     PutStr  query_msg    ; query user whether to terminate
42:     GetCh   AL           ; read response
43:     nWln
44:     cmp     AL,'Y'       ; if response is not 'Y'
45:     jne     read_char    ; read another character
46: done:
47:     .EXIT
48: main      ENDP
49:          END   main

```

Overview: 6

```

1:  TITLE    uppercase conversion of characters    TOUPPER.ASM
2:  COMMENT |
3:          Objective: To convert lowercase letters to
4:          corresponding uppercase letters.
5:          Input: Requests a character string from keyboard.
6:          Output: Prints the input string in uppercase.
7:  .MODEL SMALL
8:  .STACK 100H
9:  .DATA
10: name_prompt    DB 'Please type your name: ',0
11: out_msg        DB 'Your name in capitals is: ',0
12: in_name        DB 31 DUP (?)
13:
14:  .CODE
15:  INCLUDE io.mac
16:  main          PROC
17:              .STARTUP
18:              PutStr name_prompt    ; request character string
19:              GetStr  in_name,31    ; read input character string
20:              nwnln
21:              PutStr  out_msg
22:              mov     BX,OFFSET in_name ; BX := address of in_name

```

Overview: 7

```

23:  process_char:
24:      mov     AL,[BX]        ; move the char. to AL
25:      cmp     AL,0          ; if it is the NULL character
26:      je     done           ; conversion done
27:      cmp     AL,'a'        ; if (char < 'a')
28:      jl     not_lower_case ; not a lowercase letter
29:      cmp     AL,'z'        ; if (char > 'z')
30:      jg     not_lower_case ; not a lowercase letter
31:  lower_case:
32:      add     AL,'A'-'a'    ; convert to uppercase
33:  not_lower_case:
34:      PutCh  AL            ; write the character
35:      inc     BX           ; BX points to next char.
36:      jmp    process_char  ; go back to process next char.
37:      nwnln
38:  done:
39:      .EXIT
40:  main      ENDP
41:          END    main

```

Overview: 8

```

1:  TITLE   Add individual digits of a number   ADDIGITS.ASM
2:  COMMENT |
3:           Objective: To find the sum of individual digits of
4:           a given number. Shows character to binary
5:           conversion of digits.
6:           Input: Requests a number from keyboard.
7:           | Output: Prints the sum of the individual digits.
8:  .MODEL  SMALL
9:  .STACK  100H
10: .DATA
11: number_prompt  DB  'Please type a number (<11 digits): ',0
12: out_msg        DB  'The sum of individual digits is: ',0
13: number         DB  11 DUP (?)
14:
15: .CODE
16: INCLUDE io.mac
17: main          PROC
18:              .STARTUP

```

Overview: 9

```

19:          PutStr  number_prompt  ; request an input number
20:          GetStr  number,11      ; read input number as a string
21:          nwnln
22:          mov     BX,OFFSET number ; BX := address of number
23:          sub     DX,DX           ; DX := 0 -- DL keeps the sum
24: repeat_add:
25:          mov     AL,[BX]         ; move the digit to AL
26:          cmp     AL,0           ; if it is the NULL character
27:          je      done           ; sum is done
28:          and     AL,0FH         ; mask off the upper 4 bits
29:          add     DL,AL          ; add the digit to sum
30:          inc     BX             ; increment BX to point to next digit
31:          jmp     repeat_add      ; and jump back
32: done:
33:          PutStr  out_msg
34:          PutInt  DX             ; write sum
35:          nwnln
36:          .EXIT
37: main      ENDP
38:          END     main

```

Overview: 10