bcd to hex:

```
%macro WRITE 02
mov rax ,1
mov rdi ,1
mov rsi ,%1
mov rdx ,%2
syscall
%endmacro
%macro READ 02
mov rax ,0
mov rdi ,0
mov rsi ,%1
mov rdx ,%2
syscall
%endmacro
section .data
msg1 db "Enter the BCD no. : ",10
len1 equ $-msg1
msg2 db "Hex equavalent is : ",10
len2 equ $-msg2
section .bss
char_buff resb 17
ans resq 1
section .text
global _start
_start:
 WRITE msg1,len1
 READ char_buff,17
 dec rax
 mov rcx,rax
 mov rsi,char_buff
 mov rbx,00H
 up: mov rax,0AH
 mul rbx
 mov rbx,rax
 mov rdx,00H
 mov dl,byte[rsi]
 sub dl,30H
 add rbx,rdx
 inc rsi
 dec rcx
 jnz up
 mov [ans],rbx
 WRITE msg2,len2
 mov rbx,[ans]
 call display
exit: mov rax,60
 mov rdi,00
 syscall
display:
 mov rcx,16
 mov rsi,char_buff
 up3:rol rbx,04H
 mov dl,bl
 and dl,0FH
 cmp dl,09H
 jbe l2
 add dl,07H
l2: add dl,30H
 mov byte[rsi],dl
 inc rsi
 dec rcx
 jnz up3
 WRITE char_buff,16
 ret
```

hex to bcd:

```
%macro WRITE 02
mov rax ,1
mov rdi ,1
mov rsi ,%1
mov rdx ,%2
syscall
%endmacro
%macro READ 02
mov rax ,0
mov rdi ,0
mov rsi ,%1
mov rdx ,%2
syscall
%endmacro
section .data
msg3 db "Enter the HEX no. : ",10
len3 equ $-msg3
msg4 db "BCD equavalent is : ",10
len4 equ $-msg4
section .bss
char_buff resb 17
cnt resq 01
char resb 01
section .text
global _start
_start:
 WRITE msg3,len3
 READ char_buff,17
 call accept
 mov byte[cnt],00H
 mov rax,rbx
up1: mov rdx,00H
 mov rbx,0AH
 div rbx
 push rdx
 inc byte[cnt]
 cmp rax,00H
 jne up1
 WRITE msg4,len4
up2: pop rdx
 add dl,30H
 mov byte[char],dl
 WRITE char,01
 dec byte[cnt]
 jnz up2
exit: mov rax,60
 mov rdi,00
 syscall
accept:
 dec rax
 mov rcx,rax
 mov rsi,char_buff
 mov rbx,00H
up4: shl rbx,04H
 mov rdx,00H
 mov dl,byte[rsi]
 cmp dl,39H
 jbe l1
 sub dl,07H
l1: sub dl,30H
 add rbx,rdx
 inc rsi
 dec rcx
 jnz up4
 ret
```

addition of 2no.:

```
%macro READ 2
   mov rax, 0
   mov rdi, 0
   mov rsi, %1
   mov rdx, %2
   syscall
%endmacro
%macro WRITE 2
   mov rax, 1
   mov rdi, 1
   mov rsi, %1
   mov rdx, %2
   syscall
%endmacro
section .data
   msg1 db "Enter first number", 10
   len1 equ $-msg1
   msg2 db "Enter second number", 10
   len2 equ $-msg2
section .bss
   a resq 1
   b resq 1
   char_buff resb 16
section .text
   global _start:
_start:
   WRITE msg1, len1
   READ char_buff, 16
   dec rax
   mov rcx, rax
   call accept
   mov qword [a], rbx
   WRITE msg2, len2
   READ char_buff, 16
   dec rax
   mov rcx, rax
   call accept
   mov qword [b], rbx
   mov rbx, qword [a]
   add rbx, qword [b]
   call display
   mov rax, 60
   mov rdi, 0
   syscall
accept:
   mov rsi, char_buff
   mov rbx, 0
up: mov rdx, 0
   mov dl, byte [rsi]
      cmp dl, 39h
   jbe sub30
   sub dl, 070h
sub30:
   sub dl, 30h
   shl rbx, 4
   add rbx, rdx
   inc rsi
   dec rcx
   jnz up
   ret
display:
   mov rcx, 16
   mov rsi, char_buff
up1:
   rol rbx, 4
   mov dl, bl
   and dl, 0fh
   cmp dl, 9h
   jbe add30
   add dl, 07h
add30:
   add dl, 30h
   mov byte [rsi], dl
   inc rsi
   dec rcx
   jnz up1
   WRITE char_buff, 16
   ret
```

hello world:

```
segment .data
msg db “Hello, world!”,10
Len equ $ - msg

segment .text
global_start
_start:
mov rdi,1
mov rsi,msg
mov rdx,len
Syscall
mov rax,60
mov rdi,0
Syscall
```

multiplication(Addition):

```
section .data
msg1 db "Enter the mutiplicant",10
msg1_len equ $-msg1
msg2 db "Enter the mutiplier",10
msg2_len equ $-msg2
msg3 db "Multiplication Result:",10
msg3_len equ $-msg3
msg_space  db " ",10
msg_space_len equ $-msg_space
%macro write 2
mov rax,1
mov rdi,1
mov rsi,%1
mov rdx,%2
syscall
%endmacro
%macro read 2
mov rax,0
mov rdi,0
mov rsi,%1
mov rdx,%2
syscall
%endmacro
section .bss
num resb 17
buff resb 16
ccnt resq 1
no1 resq 1
no2 resq 1
section .text
global _start
_start:
write msg1,msg1_len
read num,17
dec rax
mov qword[ccnt],rax
call accept
mov qword[no1],rbx
write msg2,msg2_len
read num,17
dec rax
mov qword[ccnt],rax
call accept
mov qword[no2],rbx
mov rbx,00
l1:
add rbx,qword[no1]
dec qword[no2]
cmp qword[no2],00
jne l1
write msg3,msg3_len
call disp
; Exit
mov rax,60
mov rdi,0
syscall
accept:
mov rbx,00
mov rsi,num
mov rdx,00h
up1:
shl rbx,04h
mov dl,byte[rsi]
cmp dl,39h
jbe sub_30
sub dl,07h
sub_30:
sub dl,30h
add rbx,rdx
inc rsi
dec qword[ccnt]
jnz up1
ret
disp:
mov rsi,buff
mov rcx,16
mov rdx,00
up2:
rol rbx,04
mov dl,bl
and dl,0fh
cmp dl,09
jbe mc
add dl,07h
mc:
add dl,30h
mov [rsi],dl
inc rsi
dec rcx
jnz up2
write buff,16
ret
```

multiplication(Add&Shift):

```
section .data
msg1 db "Enter the mutiplicant",10
msg1_len equ $-msg1
msg2 db "Enter the mutiplier",10
msg2_len equ $-msg2
msg3 db "Multiplication Result:",10
msg3_len equ $-msg3
msg_space  db " ",10
msg_space_len equ $-msg_space
%macro write 2
mov rax,1
mov rdi,1
mov rsi,%1
mov rdx,%2
syscall
%endmacro
%macro read 2
mov rax,0
mov rdi,0
mov rsi,%1
mov rdx,%2
syscall
%endmacro
section .bss
num resb 17
buff resb 16
ccnt resq 1
B resq 1
Q resq 1
A resq 1
n resq 1
section .text
global _start
_start:
write msg1,msg1_len
read num,17
dec rax
mov qword[ccnt],rax
call acceptmov qword[B],rbx
write msg2,msg2_len
read num,17
dec rax
mov qword[ccnt],rax
call  accept
mov qword[Q],rbx
mov qword[A],00
mov qword[n],64
above:
mov rax,qword[Q]
and rax,01h
cmp rax,01h
jne shift
mov rax,qword[A]
mov rbx,qword[B]
add rax,rbx
mov qword[A],rax
shift:
mov rax,qword[A]
mov rbx,qword[Q]
shr rbx,01
and rax,01
cmp rax,01
jne shift_a
mov rdx,01
ror rdx,01
or rbx,rdx
shift_a:
mov rax,qword[A]
shr rax,01
mov qword[A],rax
mov qword[Q],rbx
dec qword[n]
jnz above
write msg3,msg3_len
mov rbx,qword[A]
call disp
mov rbx,qword[Q]
call disp
mov rax,60
mov rdi,0
syscall
accept:
mov rbx,00
mov rsi,num
mov rdx,00h
up1:
shl rbx,04h
mov dl,byte[rsi]
cmp dl,39h
jbe sub_30
sub dl,07h
sub_30:
sub dl,30h
add rbx,rdx
inc rsi
dec qword[ccnt]
jnz up1
ret
disp:
mov rsi,buff
mov rcx,16
mov rdx,00
up2:
rol rbx,04
mov dl,bl
and dl,0fh
cmp dl,09
jbe mc
add dl,07h
mc:
add dl,30h
mov [rsi],dl
inc rsi
dec rcx
jnz up2
write buff,16
ret
```

string:

```
%macro read 2
mov rax,0
mov rdi,0
mov rsi,%1
mov rdx,%2
syscall
%endmacro
%macro write 2
mov rax,1
mov rdi,1
mov rsi,%1
mov rdx,%2
syscall
%endmacro
section .data
menumsg  db 10, "1. Stringlength",10
      db "2. String copy",10
      db "3. String reverse",10
      db "4. String compare",10
      db "5. String concat",10
      db "6.Check palindrome",10
      db "7. String substring",10
      db "8.Exit",10
      db "Enter your choice -1-8",10
menulen equ $-menumsg
msg1 db "Enter String1",10
len1 equ $-msg1
msg2 db "Enter String2",10
len2 equ $-msg2
msg3 db "The length of string:",10
len3 equ $-msg3
msg4 db "The copied string",10
len4 equ $-msg4
msg5 db "The reverse  String",10
len5 equ $-msg5
msg6 db "String equal",10
len6 equ $-msg6
msg7 db "String are not equal",10
len7 equ $-msg7
msg8 db "The string concated",10
len8 equ $-msg8
msg9 db "String palindrome",10
len9 equ $-msg9
msg10 db "String not palindrome",10
len10 equ $-msg10
msg11 db "Substring ",10
len11 equ $-msg11
msg12 db "Not substring",10
len12 equ $-msg12
msg13 db "Wrong choice",10
len13 equ $-msg13
section .bss
string1 resb 20
string2 resb 20
string3 resb 40
l1 resq 1
l2 resq 1
l3 resq 1
choice resb 2
buff resb 16
char_buff resb 16
section .text
global _start
_start:
write msg1,len1
read string1,20
dec rax
mov [l1],rax
write msg2,len2
read string2,20
dec rax
mov [l2],rax
printmenu:
write menumsg,menulen
read choice,2
cmp byte[choice],31h
je strlen
cmp byte[choice],32h
je strcpy
cmp byte[choice],33h
je strrev
cmp byte[choice],34h
je strcmp
cmp byte[choice],35h
je concat
cmp byte[choice],36h
je strpal
cmp byte[choice],37h
je substr
cmp byte[choice],38h
je exit
write msg13,len13
jmp printmenu

strlen:
write msg3,len3
mov rbx,[l1]
call display
jmp printmenu

strcpy:
mov rsi,string1
mov rdi,string3
mov rcx,[l1]
cld
rep movsb
write msg4,len4
write string3,[l1]
jmp printmenu

strrev:
mov rsi,string1
add rsi,[l1]
dec rsi
mov rdi,string3
mov rcx,[l1]
up:
mov bl,byte[rsi]
mov byte[rdi],bl
dec rsi
inc rdi
dec rcx
jnz up
write msg5,len5
write string3,[l1]
jmp printmenu

strcmp:
mov rbx,[l1]
cmp rbx,[l2]
jne nonequal
mov rsi,string1
mov rdi,string2
mov rcx,[l1]
cld
repe cmpsb
jne nonequal
write msg6,len6
jmp printmenu

nonequal:
write msg7,len7
jmp printmenu

concat:
mov rsi,string1
mov rdi,string3
mov rcx,[l1]
cld
rep movsb
mov rsi,string2
mov rcx,[l2]
rep movsb
mov rbx,[l1]
add rbx,[l2]
mov [l3],rbx
write msg8,len8
write string3,[l3]
jmp printmenu

strpal:
write msg1,len1
read string1,20
dec rax
mov [l1],rax
mov rsi,string1
add rsi,[l1]
dec rsi
mov rdi,string3
mov rcx,[l1]
up1:
mov dl,byte[rsi]
mov byte[rdi],dl
dec rsi
inc rdi
dec rcx
jnz up1
mov rsi,string1
mov rdi,string3
mov rcx,[l1]
cld
repe cmpsb
jne notequal1
write msg9,len9
jmp printmenu

notequal1:
write msg10,len10
jmp printmenu

substr:
write msg1,len1
read string1,20
dec rax
mov [l1],rax
write msg2,len2
read string2,20
dec rax
mov [l2],rax
mov rbx,[l2]
mov rsi,string1
mov rdi,string2
up3:
mov al,byte[rsi]
cmp al,byte[rdi]
je same
mov rdi,string2
mov rbx,[l2]
inc rsi
jmp up3

same:
inc rsi
inc rdi
dec rbx
dec qword[l1]
cmp rbx,0
je st
cmp qword[l1],0
jne up3
write msg12,len12
jmp printmenu

st:
write msg11,len11
jmp printmenu

exit:
mov rax,60
mov rdi,00
syscall

display:
mov rsi,char_buff
mov rcx,16
up2:
rol rbx,4
mov dl,bl
and dl,0Fh
cmp dl,09h
jbe add30
add dl,07h
add30:
add dl,30h
mov byte[rsi],dl
inc rsi
dec rcx
jnz up2
write char_buff,16
ret
```