```
section .data
   msg1 db "Enter two hexadecimal ",10
   len1 equ $ - msg1

   msgAdd db 10,"Addition Result = "
   lenAdd equ $ - msgAdd

   msgDiv db 10,"Division (A / B) : ",10
   lenDiv equ $ - msgDiv

   msgQuo db "Quotient  = "
   lenQuo equ $ - msgQuo

   msgRem db "Remainder = "
   lenRem equ $ - msgRem

   newline db 10
   len_nl equ $ - newline

section .bss
   a resq 1
   b resq 1
   sum resq 1
   quotient resq 1
   remainder resq 1
   char_buff resb 17

section .text
   global _start

_start:
   WRITE msg1,len1

   READ char_buff,17
   call accept
   mov [a],rbx


   READ char_buff,17
   call accept
   mov [b],rbx

   mov rax,[a]
   add rax,[b]
   mov [sum],rax

   WRITE msgAdd,lenAdd
   mov rbx,[sum]
   call display
   WRITE newline,len_nl

   mov rdx,0
   mov rax,[a]
   div qword [b]

   mov [quotient],rax
   mov [remainder],rdx

   WRITE msgDiv,lenDiv

   WRITE msgQuo,lenQuo
   mov rbx,[quotient]
   call display
   WRITE newline,len_nl

   WRITE msgRem,lenRem
   mov rbx,[remainder]
   call display
   WRITE newline,len_nl

   EXIT


accept:
   dec rax
   mov rsi,char_buff
   xor rbx,rbx
.acc_loop:
   mov rdx,0
   mov dl,[rsi]
   cmp dl,39h
   jbe .digit
   sub dl,7h
.digit:
   sub dl,30h
   shl rbx,4
   add rbx,rdx
   inc rsi
   dec rax
   jnz .acc_loop
   ret


display:
   mov rax,16
   mov rsi,char_buff
.disp_loop:
   rol rbx,4
   mov dl,bl
   and dl,0Fh
   cmp dl,9
   jbe .add_30
   add dl,7
.add_30:
   add dl,30h
   mov [rsi],dl
   inc rsi
   dec rax
   jnz .disp_loop
   WRITE char_buff,16
   ret
```

```
%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

%macro EXIT 0
 mov rax,60
 mov rdi,1
 syscall
%endmacro

section .data

 msg1 db "Enter the HEX number : ",
 len1 equ $-msg1
 msg2 db "The BCD Equivalent : "
 len2 equ $-msg2
 msg3 db " ",10
 len3 equ $-msg3

section .bss
 char_buff resb 17
 ans resq 1
 cnt resb 1

section .text
 global _start

_start:
 WRITE msg1, len1
 READ char_buff, 17
 call accept
 WRITE msg2, len2
 mov rcx, 00
 mov rax, rbx
   l1:mov rdx, 00
 mov rbx, 0Ah
 div rbx
 push rdx
 inc rcx
 cmp rax, 00
 jnz l1
 mov byte[cnt], cl
   l2:pop rbx
 cmp bl, 09H
 jbe l3
 add bl, 07H
   l3:add bl, 30H
 mov byte[ans], bl
 WRITE ans, 01
 dec byte[cnt]
 jnz l2

 WRITE msg3, len3
 EXIT


accept:
 dec rax
 mov rsi,char_buff
 mov rbx,00
   up:mov rdx,00H
 mov dl,byte[rsi]
 cmp dl,39H
 jbe sub30
 sub dl,07H
  sub30:sub dl,30H
      shl rbx,04H
 add rbx,rdx
 inc rsi
 dec rax
 jnz up
 ret

display:
 mov rsi, char_buff
 mov rcx, 16
   up2:rol rbx, 04
 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
```

```
%macro WRITE 2
 mov rax, 01
 mov rdi, 01
 mov rsi, %1
 mov rdx, %2
 syscall
%endmacro
%macro READ 2
      mov rax, 00
      mov rdi, 00
      mov rsi, %1
      mov rdx, %2
      syscall
%endmacro
%macro EXIT 00
 mov rax, 60
 mov rdi, 00
 syscall
%endmacro

section .data

 msg1 db "Enter two numbers : ",10
 len1 equ $-msg1
 msg2 db "Multiplication by Shift & Add : "
 len2 equ $-msg2
 msg3 db " ",10
 len3 equ $-msg3


section .bss
 char_buff resb 17
 actl resq 1
 m resq 1
 n resq 1
 ans resq 1
 A resq 1
 B resq 1
 Q resq 1

section .text
 global _start

_start:
      WRITE msg1, len1
 READ char_buff, 17
 call accept
 mov [m], rbx
 READ char_buff ,17
 call accept
 mov [n], rbx


 mov qword[A], 00H
 mov rbx, qword[m]
 mov qword[B], rbx
 mov rbx, qword[n]
 mov qword[Q], rbx
 mov rcx, 64
   l2:mov rbx, [Q]
 AND rbx, 01H
 jz ShiftAQ
 mov rbx, [B]
 add [A], rbx
ShiftAQ:shr qword [Q], 01H
 mov rbx, [A]
 AND rbx, 01
 jz ShiftA
 mov rbx, 01
 ror rbx, 01
 OR [Q], rbx
ShiftA:shr qword[A], 01
 dec rcx
 jnz l2
 WRITE msg2, len2
 mov rbx, [A]
 call display
 mov rbx, [Q]
 call display

 WRITE msg3, len3
 EXIT

accept:
 dec rax
 mov rsi,char_buff
 mov rbx,00
   up:mov rdx,00H
 mov dl,byte[rsi]
 cmp dl,39H
 jbe sub30
 sub dl,07H
  sub30:sub dl,30H
      shl rbx,04H
 add rbx,rdx
 inc rsi
 dec rax
 jnz up
 ret

display:
 mov rax,16
 mov rsi,char_buff
  above:rol rbx,04H
 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 rax
 jnz above
 WRITE char_buff,16
 ret

 inc rsi
 dec rcx
 jnz above
 WRITE char_buff, 16
 ret
```

```
%macro WRITE 2
 mov rax, 01
 mov rdi, 01
 mov rsi, %1
 mov rdx, %2
 syscall
%endmacro


%macro READ 2
      mov rax, 00
      mov rdi, 00
      mov rsi, %1
      mov rdx, %2
      syscall
%endmacro

%macro EXIT 00
 mov rax, 60
 mov rdi, 00
 syscall
%endmacro

section .data

 msg1 db "Enter two numbers : ",10
 len1 equ $-msg1
 msg2 db "Multiplication by Successive Addition : "
 len2 equ $-msg2
 msg3 db " ",10
 len3 equ $-msg3

section .bss
 char_buff resb 17
 actl resq 1
 m resq 1
 n resq 1
 ans resq 1

section .text
 global _start

_start:
      WRITE msg1, len1
 READ char_buff, 17
 call accept
 mov [m], rbx
 READ char_buff ,17
 call accept
 mov [n], rbx


 mov rbx, 00H
 mov rcx, [n]
    l1:add rbx, [m]
 dec rcx
 jnz l1
 mov [ans], rbx
 WRITE msg2, len2
 mov rbx, [ans]
 call display

 WRITE msg3, len3
 EXIT

accept:
 dec rax
 mov rsi,char_buff
 mov rbx,00
    up:mov rdx,00H
 mov dl,byte[rsi]
 cmp dl,39H
 jbe sub30
 sub dl,07H
  sub30:sub dl,30H
       shl rbx,04H
 add rbx,rdx
 inc rsi
 dec rax
 jnz up
 ret

display:
 mov rax,16
 mov rsi,char_buff
  above:rol rbx,04H
 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 rax
 jnz above
 WRITE char_buff,16
 ret

 inc rsi
 dec rcx
 jnz above
 WRITE char_buff, 16
 ret
```

```
%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
%macro EXIT 0
 mov rax,60
 mov rdi,1
 syscall
%endmacro
section .data

 msg1 db "Enter the BCD number : "
 len1 equ $-msg1
 msg2 db "The Hex Equivalent : "
 len2 equ $-msg2
 msg3 db " ",10
 len3 equ $-msg3
section .bss
 char_buff resb 17
 ans resq 1

section .text
 global _start

_start:
 WRITE msg1, len1
 READ char_buff, 17
 dec rax
 mov rbx, 00
 mov rsi, char_buff
 mov rcx, rax
   up3: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 up3
 mov [ans], rbx
 WRITE msg2, len2
 mov rbx, [ans]
 call display

 WRITE msg3, len3
 EXIT
accept:
 dec rax
 mov rsi,char_buff
 mov rbx,00
    up:mov rdx,00H
 mov dl,byte[rsi]
 cmp dl,39H
 jbe sub30
 sub dl,07H
  sub30:sub dl,30H
       shl rbx,04H
 add rbx,rdx
 inc rsi
 dec rax
 jnz up
 ret
display:
 mov rsi, char_buff
 mov rcx, 16
   up2:rol rbx, 04
 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
```

```
%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

%macro EXIT 0
   mov rax,60
   xor rdi,rdi
   syscall
%endmacro

section .data
   msg1 db "Enter two hexadecimal ",10
   len1 equ $ - msg1

   msgSub db 10,"Subtraction (A - B) = "
   lenSub equ $ - msgSub

   msgMul db 10,"Multiplication Result (A * B):",10
   lenMul equ $ - msgMul

   msgHigh db "High 64 bits : "
   lenHigh equ $ - msgHigh

   msgLow db "Low 64 bits  : "
   lenLow equ $ - msgLow

   newline db 10
   len_nl equ $ - newline

section .bss
   a resq 1
   b resq 1
   sub_res resq 1
   mul_high resq 1
   mul_low resq 1
   char_buff resb 17

section .text
   global _start

_start:
   WRITE msg1,len1
   READ char_buff,17
   call accept
   mov [a],rbx
   READ char_buff,17
   call accept
   mov [b],rbx
   mov rax,[a]
   sub rax,[b]
   mov [sub_res],rax

   WRITE msgSub,lenSub
   mov rbx,[sub_res]
   call display
   WRITE newline,len_nl

   WRITE msgMul,lenMul
   mov rax,[a]
   mul qword [b]
   mov [mul_high],rdx
   mov [mul_low],rax

   WRITE msgHigh,lenHigh
   mov rbx,[mul_high]
   call display
   WRITE newline,len_nl

   WRITE msgLow,lenLow
   mov rbx,[mul_low]
   call display
   WRITE newline,len_nl

   EXIT

accept:
   dec rax
   mov rsi,char_buff
   xor rbx,rbx
.acc_loop:
   mov rdx,0
   mov dl,[rsi]
   cmp dl,39h
   jbe .digit
   sub dl,7h
.digit:
   sub dl,30h
   shl rbx,4
   add rbx,rdx
   inc rsi
   dec rax
   jnz .acc_loop
   ret

display:
   mov rax,16
   mov rsi,char_buff
.disp_loop:
   rol rbx,4
   mov dl,bl
   and dl,0Fh
   cmp dl,9
   jbe .add30
   add dl,7
.add30:
   add dl,30h
   mov [rsi],dl
   inc rsi
   dec rax
   jnz .disp_loop
   WRITE char_buff,16
   ret
```

```
%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

%macro EXIT 0
   mov rax,60
   xor rdi,rdi
   syscall
%endmacro

section .data
   menu db 10,"1. String Length",10,\
         "2. String Compare",10,\
         "3. String Copy",10,\
         "4. Exit",10,"Enter choice: ",0
   menulen equ $-menu

   msg1 db "Enter string: ",0
   len1 equ $-msg1

   msg2 db "Enter first string: ",0
   len2 equ $-msg2

   msg3 db "Enter second string: ",0
   len3 equ $-msg3

   msg_equal db "Strings are Equal",10,0
   len_equal equ $-msg_equal

   msg_notequal db "Strings are NOT Equal",10,0
   len_notequal equ $-msg_notequal

   msg_copied db "Copied String: ",0
   len_copied equ $-msg_copied

   msg_len db "Length = ",0
   len_len equ $-msg_len

section .bss
   str1 resb 128
   str2 resb 128
   outbuf resb 8
   choice resb 2

section .text
global _start

_start:
menu_start:
   WRITE menu, menulen
   READ choice,2

   cmp byte [choice],'1'
   je strlen_op

   cmp byte [choice],'2'
   je strcmp_op

   cmp byte [choice],'3'
   je strcpy_op

   cmp byte [choice],'4'
   je end_prog

   jmp menu_start

strlen_op:
   WRITE msg1,len1
   READ str1,128

   mov rsi,str1
   xor rcx,rcx

.len_loop:
   mov al,[rsi]
   cmp al,10
   je .len_done
   inc rcx
   inc rsi
   jmp .len_loop

.len_done:
   mov rbx,rcx
   mov rdi,outbuf+7
   mov byte [rdi],0
   dec rdi

.make_ascii:
   xor rdx,rdx
   mov rax,rbx
   mov rcx,10
   div rcx
   add dl,'0'
   mov [rdi],dl
   mov rbx,rax
   dec rdi
   test rax,rax
   jnz .make_ascii
   inc rdi

   WRITE msg_len,len_len
   WRITE rdi,8
   jmp menu_start

strcmp_op:
   WRITE msg2,len2
   READ str1,128

   WRITE msg3,len3
   READ str2,128
```

```
%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

%macro EXIT 0
   mov rax,60
   xor rdi,rdi
   syscall
%endmacro

section .data
   menu db 10,"1. String Length",10,\
                  "2. String Concatenation",10,\
                  "3. String Palindrome",10\
                  ,"4. Exit",10,"Enter choice: ",0
   menulen equ $-menu

   msg1 db "Enter string: ",0
   len1 equ $-msg1

   msg2 db "Enter first string: ",0
   len2 equ $-msg2

   msg3 db "Enter second string: ",0
   len3 equ $-msg3

   msg_len db "Length = ",0
   len_len equ $-msg_len

   msg_pal1 db "Palindrome",10,0
   len_pal1 equ $-msg_pal1

   msg_pal2 db "Not Palindrome",10,0
   len_pal2 equ $-msg_pal2

   newline db 10
   len_nl equ 1

section .bss
   str1 resb 128
   str2 resb 128
   outbuf resb 16
   choice resb 2

section .text
global _start

_start:
main_menu:
   WRITE menu, menulen
   READ choice,2

   cmp byte [choice], '1'
   je strlen_op

   cmp byte [choice], '2'
   je concat_op

   cmp byte [choice], '3'
   je palin_op

   cmp byte [choice], '4'
   je done

   jmp main_menu

strlen_op:
   WRITE msg1, len1
   READ str1,128

   mov rsi, str1
   xor rcx, rcx

.strlen_loop:
   mov al, [rsi]
   cmp al, 10
   je .done_len
   inc rcx
   inc rsi
   jmp .strlen_loop

.done_len:
   ; convert RCX -> ASCII
   mov rbx, rcx
   mov rdi, outbuf+15
   mov byte [rdi],0
   dec rdi

.convert:
   xor rdx, rdx
   mov rax, rbx
   mov rcx, 10
   div rcx
   add dl,'0'
   mov [rdi],dl
   mov rbx,rax
   dec rdi
   test rax,rax
   jnz .convert
   inc rdi

   WRITE msg_len, len_len
   WRITE rdi, 16
   WRITE newline, len_nl

   jmp main_menu

concat_op:
   WRITE msg2, len2
   READ str1,128
```

```
%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

%macro EXIT 0
   mov rax,60
   xor rdi,rdi
   syscall
%endmacro

section .data
   menu db 10,"1. String Length",10,\
         "2. String Reverse",10,\
         "3. String Palindrome",10,\
         "4. Exit",10,\
         "Enter choice: ",0
   menulen equ $-menu

   msg1 db "Enter string: ",0
   len1 equ $-msg1

   msg_len db "Length = ",0
   len_len equ $-msg_len

   msg_rev db "Reversed String: ",0
   len_rev equ $-msg_rev

   msg_pal db "Palindrome",10,0
   len_pal equ $-msg_pal

   msg_notpal db "Not Palindrome",10,0
   len_notpal equ $-msg_notpal

section .bss
   str1 resb 128
   str2 resb 128
   numbuf resb 8
   choice resb 2

section .text
global _start

_start:
menu_start:
   WRITE menu,menulen
   READ choice,2

   cmp byte[choice],'1'
   je strlen_op

   cmp byte[choice],'2'
   je strrev_op

   cmp byte[choice],'3'
   je palin_op

   cmp byte[choice],'4'
   je exit_now

   jmp menu_start

strlen_op:
   WRITE msg1,len1
   READ str1,128

   mov rsi,str1
   xor rcx,rcx

.len_loop:
   mov al,[rsi]
   cmp al,10
   je .done_len
   inc rcx
   inc rsi
   jmp .len_loop

.done_len:
   mov rbx,rcx
   mov rdi,numbuf+7
   mov byte[rdi],0
   dec rdi
.make_ascii:
   xor rdx,rdx
   mov rax,rbx
   mov rcx,10
   div rcx
   add dl,'0'
   mov [rdi],dl
   mov rbx,rax
   dec rdi
   test rax,rax
   jnz .make_ascii
   inc rdi

   WRITE msg_len,len_len
   WRITE rdi,8
   jmp menu_start

   WRITE msg1,len1
   READ str1,128
   mov rsi,str1
   xor rcx,rcx

.rev_len:
   mov al,[rsi]
   cmp al,10
   je .got_len
   inc rcx
   inc rsi
   jmp .rev_len
```

```
.got_len:
 mov rsi,str1
 mov rdi,str1
 add rdi,rcx
 dec rdi
 mov rbx,0

.rev_copy:
 cmp rdi,str1
 jb .done_rev
 mov al,[rdi]
 mov [str2+rbx],al
 dec rdi
 inc rbx
 jmp .rev_copy

.done_rev:
 mov byte[str2+rbx],10

 WRITE msg_rev,len_rev
 WRITE str2,128

 jmp menu_start
palin_op:
 WRITE msg1,len1
 READ str1,128

 mov rsi,str1
 xor rcx,rcx

.p1:
 mov al,[rsi]
 cmp al,10
 je .p_len_done
 inc rcx
 inc rsi
 jmp .p1

.p_len_done:
 mov rdi,str1
 mov rsi,str1
 add rsi,rcx
 dec rsi

.check_pal:
 cmp rdi,rsi
 jge .pal_yes
 mov al,[rdi]
 mov bl,[rsi]
 cmp al,bl
 jne .pal_no
 inc rdi
 dec rsi
 jmp .check_pal

.pal_yes:
 WRITE msg_pal,len_pal
 jmp menu_start

.pal_no:
 WRITE msg_notpal,len_notpal
 jmp menu_start

exit_now:
 EXIT
```

```
 WRITE msg3, len3
 READ str2,128

 mov rsi, str1
.find_end:
 cmp byte [rsi],10
 je .start_copy
 inc rsi
 jmp .find_end

.start_copy:
 mov rdi, str2
.copy_loop:
 mov al,[rdi]
 mov [rsi],al
 cmp al,10
 je .done_copy
 inc rsi
 inc rdi
 jmp .copy_loop

.done_copy:
 WRITE str1,128
 jmp main_menu

palin_op:
 WRITE msg1, len1
 READ str1,128

 mov rsi,str1
 xor rcx,rcx
.p_loop1:
 mov al,[rsi]
 cmp al,10
 je .have_len
 inc rcx
 inc rsi
 jmp .p_loop1

.have_len:
 mov rdi,str1
 mov rsi,str1
 add rsi,rcx
 dec rsi

.check_loop:
 cmp rdi,rsi
 jge .pal_yes
 mov al,[rdi]
 mov bl,[rsi]
 cmp al,bl
 jne .pal_no
 inc rdi
 dec rsi
 jmp .check_loop

.pal_yes:
 WRITE msg_pal1,len_pal1
 jmp main_menu

.pal_no:
 WRITE msg_pal2,len_pal2
 jmp main_menu


done:
 EXIT
```

```
 mov rsi,str1
 mov rdi,str2
.compare_loop:
 mov al,[rsi]
 mov bl,[rdi]
 cmp al,10
 je .end_check
 cmp bl,10
 je .end_check
 cmp al,bl
 jne .notequal
 inc rsi
 inc rdi
 jmp .compare_loop
.end_check:
 cmp al,bl
 jne .notequal
 WRITE msg_equal,len_equal
 jmp menu_start
.notequal:
 WRITE msg_notequal,len_notequal
 jmp menu_start
strcpy_op:
 WRITE msg1,len1
 READ str1,128
 mov rsi,str1
 mov rdi,str2
.copy_loop:
 mov al,[rsi]
 mov [rdi],al
 cmp al,10
 je .copy_done
 inc rsi
 inc rdi
 jmp .copy_loop
.copy_done:
 WRITE msg_copied,len_copied
 WRITE str2,128
 jmp menu_start
end_prog:
 EXIT
```