Microprocessor Experiment No. 1
Name: Swayam Shinde
Class: SE COMP-B
Roll No: 22CO114

Write an X86/64 ALP to accept 5 64-bit hexadecimal no's for user and store them in an array and display the nos.

```
msg1 db "Enter 5 64-bit numbers ",10,13
len1 equ $-msg1
msg2 db "Entered 5 64-bit numbers are
",10,13
len2 equ $-msg2
section .bss
array resb 120
count resb 1
section .text
global _start
_start: ;logic to read 5 64-bit numbers
mov bh,05 ; count is in bh
mov rbx,00 ; array index
up: mov rax,00
mov rdi,00
mov rsi,array
add rsi,rbx
mov rdx,17
syscall
add rbx,17
dec byte[count] ; dec count
jnz up ; check count is 0 or not using zero flag
i;
mov byte[count],05 ; count is in bh
mov rbx,00 ;array index
up1: mov rax,01
mov rdi,01
mov rsi,array
add rsi,rbx
mov rdx,17
syscall
add rbx,17
dec byte[count] ; dec count
jnz up1 ; check count is 0 or not using zero flag
i;

;exit syscall
mov rax, 60
mov rdi,00
syscall
```

OUTPUT

```
[root@localhost nasm-2.11.02rc6]# nasm -f elf64 mp_1.asm
[root@localhost nasm-2.11.02rc6]# ld mp_1.o -o hn
[root@localhost nasm-2.11.02rc6]# ./hn
Enter 5 64-bit numbers
12345a65432
837928373
7283629
8363628
1637b83739
Entered 5 64-bit numbers are
12345A65432
837928373
7283629
8363628
1637B83739
```

Microprocessor Experiment No. 2
Name: Swayam Shinde
Class: SE COMP-B
Roll No: 22CO114

Write an X86/64 ALP to accept a string and display its length.

```
section .data
    msg db 10,'Enter the string :',10
    msg_len equ $-msg
    msg1 db 10,'Length of string in Hex = '
    msg1_len equ $-msg1
section.bss
   dispbuff resb 2
   buf resb 50
   buf_len: equ $-buf
   len resb 1
%macro print   2
  mov rax,4
  mov rbx,1
  mov rcx,%1
  mov rdx,%2
   int 0*80
%endmacro
%macro accept 2
  macro rax,3
  macro rbx,0
  macro rcx,%1
  macro rdx,%2

  int 80h
%endmacro
section .text
  global_start
_start:
  print msg,msg_len
  accept buf,buf_len
  dec al
  dec al
  mov [len],al
  print msg1,msg1_len
  mov bl, [len]
  call disp8num
exit:
  mov rax,1
  mov rbx,0
  int 0*80
disp8num:
  mov rdi,dispbuff
  mov rcx,2
dispup1:
  rol bl,4
  mov dl,bl
  and dl,0fh
  add dl,30h
  cmp dl,39h
  jbe dispskip1
  add dl,07h
dispskip1:
  mov [rdi],dl

  inc rdi
  loop dispup1
  print dispbuff,2
  ret
```

OUTPUT

```
[root@localhost nasm-2.11.02rc6]# nasm -f elf64 string.asm
[root@localhost nasm-2.11.02rc6]# ld string.o -o str
[root@localhost nasm-2.11.02rc6]# ./str
Enter the string : Microprocessor
Length of string in Hex = 0E
[root@localhost nasm-2.11.02rc6]#
```

Microprocessor Experiment No. 3
Name: Swayam Shinde
Class: SE COMP-B
Roll No: 22CO114

Write a X86/64 ALP to find the largest of given byte/word/Dword/64-bit numbers.

```
section .data
array db 11h, 55h, 33h, 22h,44h
msg1 db 10,13,&amp;quot;Largest no in an array is:&amp;quot;
len1 equ $-msg1
section .bss
cnt resb 1
result resb 16
section .text
global _start
_start:
mov byte[cnt],5
mov rsi,array
mov al,0
LP: cmp al,[rsi]
jg skip
xchg al ,[rsi]
skip: inc rsi
dec byte[cnt]
jnz LP

;display
mov Rax,1

mov Rdi,1
mov Rsi,msg1
mov Rdx,len1
syscall
;display al
call display
;exit system call
mov Rax ,60
mov Rdi,0
syscall
```

```
%macro dispmsg 2
mov Rax,1
mov Rdi,1
mov rsi,%1
mov rdx,%2
syscall
%endmacro
display:
mov rbx,rax              ; store no in rbx
mov rdi,result           ;point rdi to result variable
mov cx,16                ;load count of rotation in cl
up1:
rol rbx,04          ;rotate no of left by four bits
mov al,bl       ; move lower byte in dl
and al,0fh          ;get only LSB
cmp al,09h          ;compare with 39h

jg add_37           ;if greater than 39h skip add 37
add al,30h
jmp skip1           ;else add 30
add_37:
add al,37h
skip1:
mov [rdi],al        ;store ascii code in result variable
inc rdi             ; point to next byte
dec cx              ; decrement counter
jnz up1             ; if not zero jump to repeat
dispmsg result,16       ;call to macro

ret
```

OUTPUT:

Largest no in an array is:000000000000001C

Microprocessor Experiment No. 5
Name: Swayam Shinde
Class: SE COMP-B
Roll No: 22CO114

Write an X86/64 ALP to count the number of positive and negative numbers from the array.

```
section .data
nline db 10,10
nline_len equ  $-nline
arr dd  -11111111H, 22222222H, -33333333H,
-44444444H, -55555555H
arr_size equ 5
pmsg db   10,10,&amp;quot;The no. of Positive
elements in 32-bit array:&amp;quot;
pmsg_len equ $-pmsg
nmsg  db  10,10,&amp;quot;The no. of Negative
elements in 32-bit array:&amp;quot;
nmsg_len equ $-nmsg;
section  .bss
p_count  resq   01
n_count  resq   01
dnumbuff resb   02
%macro display 2
   mov rax,01
   mov rdi,01
   mov rsi,%1
   mov rdx,%2
   syscall
%endmacro
section .text
global   _start
_start:
   mov    esi, arr
   mov    ecx,5        ;Arraay counter i.e.5
   mov    ebx,0;       ; counter for     +ve nos.
   mov    edx,0;       ; counter for    -ve nos.
next_num:
   mov    eax,[esi]    ; take no. in RAX
   rcl    eax,1        ; rotate left 1 bit to check for
sign bit
   jc    negative
positive:
   inc    ebx          ; no carry, so no. is +ve
   jmp    next
negative:
   inc    edx          ; carry, so no. is -ve
next:
   add  esi,4              ; 32 bit nos i.e. 4 bytes
   loop next_num
   mov [p_count], ebx       ; store positive
count
   mov [n_count], edx       ; store negative
count
   display pmsg, pmsg_len
   mov ebx,[p_count]        ; load value of
p_count in rax
   call disp8_proc          ; display p_count
   display nmsg, nmsg_len
   mov ebx,[n_count]        ; load value of
n_count in rax
   call  disp8_proc         ; display n_count
   display    nline, nline_len
 exit:
   mov rax,60       ;Exit
   mov rbx,00
   syscall
disp8_proc:
   mov edi,dnumbuff    ;point edi to buffer
   mov ecx,02       ;load number of digits to
display
dispup1:
   rol bl,4       ;rotate number left by four bits
   mov dl,bl      ;move lower byte in dl
   and dl,0fh     ;mask upper digit of byte in dl
   add dl,30h     ;add 30h to calculate ASCII
code
   cmp dl,39h     ;compare with 39h
   jbe dispskip1   ;if less than 39h akip adding
07 more
   add dl,07h     ;else add 07
dispskip1:
   mov [edi],dl    ;store ASCII code in buffer
   inc edi         ;point to next byte
   loop dispup1    ;decrement the count of
digits to display

                  ;if not zero jump to repeat
   display dnumbuff,2
   ret
```

OUTPUT

```
[root@localhost A1]# nasm -f elf64 A1.asm
[root@localhost A1]# ld -o A1 A1.o
[root@localhost A1]# ./A1

The no. of Positive elements in 32-bit array :    01
The no. of Negative elements in 32-bit array :    04
```

Microprocessor Experiment No. 7
Name: Swayam Shinde
Class: SE COMP-B
Roll No: 22CO114

Write X86/54 ALP to detect protected mode and display the values of GDTR, LDTR, IDTR, TR and MSW Registers also identify CPU type using CPUID instruction.

```
section .data

rmsg db 10,'Processor is in real mode'
rmsg_len:equ $-rmsg
pmsg db 10,'Processor is in protected mode'
pmsg_len:equ $-pmsg
gmsg db 10,'GDT contents are : '
gmsg_len:equ $-gmsg
lmsg db 10,'LDT contents are : '
lmsg_len:equ $-lmsg
imsg db 10,'IDT contents are : '
imsg_len:equ $-imsg
tmsg db 10,'Task Register contents are : '
tmsg_len:equ $-tmsg
mmsg db 10,'Machine Status Word : '
mmsg_len:equ $-mmsg
colmsg db ':'
newline db 10
section .bss
gdt resd 1
resw 1
ldt resw 1
idt resd 1
resw 1
tr resw 1
cr0_data resd 1
dnum_buff resb 04
%macro disp 2

%endmacro
section .text

mov rax,1
mov rdi,1
mov rsi,%1
mov rdx,%2
syscall

global _start
_start:
smsw eax
mov [cr0_data],rax
bt eax,0
jc prmode
disp rmsg,rmsg_len
jmp nxt1
prmode : disp pmsg,pmsg_len
nxt1 :

sgdt [gdt]
sldt [ldt]
sidt [idt]

str[tr]dispgmsg,gmsg_lenmovbx,[gdt+4]

exit :

disp_num :
up1:

skip1:

ret

call disp_num
mov bx,[gdt+2]
call disp_num
disp colmsg,1
mov bx,[gdt]
call disp_num
disp lmsg,lmsg_len
mov bx,[ldt]
call disp_num
disp imsg,imsg_len
mov bx,[idt+4]
call disp_num
mov bx,[idt+2]
call disp_num
disp colmsg,1
mov bx,[idt]
call disp_num
disp tmsg,tmsg_len
mov bx,[tr]
call disp_num
```

```
disp mmsg,mmsg_len
mov bx,[cr0_data+2]
call disp_num
mov bx,[cr0_data]
call disp_num
disp newline,1
mov eax,01
mov ebx,00
int 80h
mov esi,dnum_buff
mov ecx,04
rol bx,4
mov dl,bl
and dl,0fh
add dl,30h
cmp dl,39h
jbe skip1
add dl,07h
mov [esi],dl
inc esi
loop up1
disp dnum_buff,4
```

OUTPUT

```
[root@localhost nasm-2.11.02rc6]# nasm -f elf64 protected_mode.asm
[root@localhost nasm-2.11.02rc6]# ld protected_mode.o -o pb
[root@localhost nasm-2.11.02rc6]# ./pb
Processor is in protected mode
GDT contents are : FF576000:007F
LDT contents are : 0000
IDT contents are : FF57B000:0FFF
Task Register contents are : 0040
Machine Status Word : 8005FFFF
[root@localhost nasm-2.11.02rc6]#
```

Microprocessor Experiment No. 8
Name: Swayam Shinde
Class: SE COMP-B
Roll No: 22CO114

Write X86/64 ALP to perform non-overlapped block transfer (with and without string specific instructions). Block containing data can be defined in the data segment.

```
section .data
msg db 10,10,&amp;#39;---Menu for Non-overlapped BLock Transfer---&amp;#39;,10
msg_lenequ $-msg

blk_bfrmsg db 10,&amp;#39;Block contents before transfer::&amp;#39;
blk_bfrmsg_lenequ $-blk_bfrmsg
blk_afrmsg db 10,&amp;#39;Block contents after transfer::&amp;#39;
blk_afrmsg_lenequ $-blk_afrmsg
srcblkdb 01h,02h,03h,04h,05h,06h,07h,00h,00h,00h,00h,00h,00h,00h
cntequ 07
spacechardb 20h
ifmsgdb 10,10
section .bss
optionbuffresb 02
dispbuffresw 02
%macro dispmsg 2
mov rax,01
mov rdi,01
mov rsi,%1
mov rdx,%2
syscall
%endmacro
%macro accept 2
mov rax,0
mov rdi,0
mov rsi, %1

mov rdx, %2
syscall
%endmacro
section .text
global _start
_start:
dispmsg blk_bfrmsg, blk_bfrmsg_len
calldispblk_proc

accept optionbuff,02
call blkxferwo_proc

ext: mov rax,60
mov rbx,0
syscall
dispblk_proc:
movrsi,srcblk
movrcx, cnt
addrcx, rcx
rdisp:
pushrcx
movbl, [rsi]
pushrsi
call disp8_proc
poprsi
incrsi
pushrsi
dispmsg spacechar,1
poprsi
poprcx

looprdisp
ret
blkxferwo_proc:
mov rsi, srcblk+6
movrdi,rsi
movrcx,cnt
addrdi, rcx
blkup1:
mov al, [rsi]
mov [rdi], al
decrsi
decrdi
loop blkup1
ret
disp8_proc:
mov cl,2
movrdi, dispbuff
dup1:
rol bl,4
movdl, bl
```

```
and dl,0fh
cmp dl,09
jbed skip
add dl,07h
dskip: add dl,30h
mov [rdi], dl
incr di
loop dup1
disp msg dispbuff,2
ret
```

OUTPUT:

```
[root@localhost ~]# cdnasm
[root@localhostnasm]# nasm -f elf64 block.asm
[root@localhostnasm]# ldblock.o -o pb
[root@localhostnasm]# ./pb
Block contents before transfer::01 02 03 04 05 06 07 00 00 00 00 00 00 00

Block contents after transfer::01 02 03 04 05 06 07 01 02 03 04 05 06 07
```

Microprocessor Experiment No. 9
Name: Swayam Shinde
Class: SE COMP-B
Roll No: 22CO114

Write an X86/64 ALP to perform overlapped block transfer with string specific instructions. Block containing data can be defined in the data segment.

```
section .data
menumsgdb 10,10,&#39;.....menu for overlap
block transfer.....&#39;,10

menumsg_lenequ $-menumsg

blk_bfrmsgdb 10,&#39;block contents before
transfer:&#39;
blk_bfrmsg_lenequ $-blk_bfrmsg
blk_afrmsgdb 10,&#39;block contents after
transfer:&#39;
blk_afrmsg_lenequ $-blk_afrmsg
positiondb 10,&#39;enter position
overlap:&#39;
pos_lenequ $-position
srcblkdb
01h,02h,03h,04h,05h,00h,00h,00h,00h,00h
cntequ 05
spacechardb 20h
ifmsgdb 10,10
section .bss
optionbuffresb 02
dispbuffresw 02
posresb 00
%macro dispmsg 2
mov rax,1
mov rdi,1
mov rsi,%1
mov rdx,%2
syscall
%endmacro
%macro accept 2
mov rax,0
mov rdi,0

mov rsi,%1
mov rdx,%2
syscall
%endmacro

section .text
global _start
_start:
dispmsgblk_bfrmsg,blk_bfrmsg_len
call dispblk_proc
dispmsg position ,pos_len
accept optionbuff,02
callpacknum_proc
callblkxferwo_proc
jmp exit1

ext: mov rax,60
mov rbx,0
syscall
dispblk_proc:
movrsi,srcblk
movrcx,cnt
addrcx,[pos]
rdisp:
pushrcx
movbl,[rsi]
pushrsi
call disp8_proc
poprsi
incrsi
pushrsi
dispmsg spacechar,1
poprsi
poprcx
looprdisp
ret
blkxferwo_proc:
mov rsi,srcblk+4
movrdi,rsi
addrdi,[pos]
movrcx,cnt

blkup1:
mov al,[rsi]
mov[rdi],al
decrsi
decrdi
```

```
loop blkup1
ret
blkxferw_proc:
mov rsi,srcblk+4
movrdi,rsi
addrdi,[pos]
movrcx,cnt
std
repmovsb
ret
disp8_proc:
mov cl,2
movrdi,dispbuff
dup1:
rol bl,4
movdl,bl
and dl,0fh
cmp dl,09h
jbedskip
add dl,07h
dskip:
add dl,30h
mov[rdi],dl
incrdi
loop dup1
dispmsg dispbuff,2
ret
packnum_proc:
movrsi,optionbuff
movbl,[rsi]
cmp bl,39h
jbe skip1
sub bl,07h
skip1:
sub bl,30h
mov[pos],bl
ret
```

OUTPUT:

```
[root@localhost nasm-2.11]# nasm -f elf64 overlap.asm
[root@localhost nasm-2.11]# ldoverlap.o -o pb
[root@localhost nasm-2.11]# ./pb
Block Contents Before Transfer : 01 02 03 04 05

Block Contents Before Transfer : 01 02 03 04 05
```

Microprocessor Experiment No. 11
Name: Swayam Shinde
Class: SE COMP-B
Roll No: 22CO114

Write X86 Assembly Language Program (ALP) to implement following OS commands.
i) COPY, ii) TYPE

```
%macro print 2
mov rax,1
mov rdi,1
mov rsi,%1
mov rdx,%2
syscall
%endmacro
%macro accept 2
mov rax,0
mov rdi,0
mov rsi,%1
mov rdx,%2
syscall
%endmacro
%macro iomodule 4
mov rax,%1
mov rdi,%2
mov rsi,%3
mov rdx,%4
syscall
%endmacro
%macro exit 0
mov rax,60
syscall
%endmacro
section .data
   cmd db 10,13, "command menu"
      db 10,13,"1.TYPE"
      db 10,13,"2.COPY"
      db 10,13,"3.DELETE"
      db 10,13,"4.EXIT"
      db 10,13,"Enter choice:"
   len equ $ – cmd
   entercmd db 10,13,"Enter command:——: "
   cmdlen equ $ – entercmd
   msg0 db 10,13,"Failed to open the file!!!"
   len0 equ $ – msg0
   msg1 db 10,13,"File opened successfully!!!"
   len1 equ $ – msg1

   msg2 db 10,13,"Failed to open the file!!!"
   len2 equ $ – msg2
   msg3 db 10,13, "Command not found!!!!"
   len3 equ $ – msg3
section .bss
   buffer resb 200
   bufferlen resb 8
   choice resb 50
   chlen equ $ – choice
   fdis1 resb 200
   fdis2 resb 200
   cnt1 resb 2
   ch1 resb 2
   name1 resb 20
   name2 resb 20
   size resb 200
section .text
global _start
_start:
   print cmd, len
   accept ch1, 2
   mov rsi, ch1
   cmp byte[rsi], '1'
   je TYPE
   cmp byte[rsi], '2'
   je COPY
   cmp byte[rsi], '3'
   je DELETE
   cmp byte[rsi], '4'
   exit
   jmp error
TYPE:
   print entercmd, cmdlen
   accept choice, chlen
   mov rsi, choice
   mov byte[size], al
   dec byte[size]
   mov al, byte[rsi]
nd:
   cmp al, 't'
   jne error
   inc rsi
   dec byte[size]
   mov al, byte[rsi]
```

```
    cmp al, 'y'
    jne error
    inc rsi

    dec byte[size]
    mov al, byte[rsi]
    cmp al, 'p'
    jne error
    inc rsi
    dec byte[size]
    mov al, byte[rsi]
    cmp al, 'e'
    jne error
    inc rsi
    dec byte[size]
    jmp success
error:
    print msg3, len3
    jmp _start
success:
    inc rsi
    dec byte[size]
    mov rdi, name1
top:
    mov al, byte[rsi]
    mov [rdi], al
    inc rdi
    inc rsi
    dec byte[size]
    jnz top
    iomodule 2, name1, 2, 777
    mov qword[fdis1], rax
    bt rax, 63
    jc error
    iomodule 0, [fdis1], buffer, 200
    mov qword[cnt1], rax
    iomodule 1, 1, buffer, qword[cnt1]
    mov rax, 3
    mov rdi, name1
    syscall
    jmp _start
DELETE:
    print entercmd, cmdlen
    accept choice, chlen
    mov byte[size], al
    dec byte[size]
    mov al, byte[rsi]
nd1:

    cmp al, 'd'
    jne error
    inc rsi
    dec byte[size]
    mov al, byte[rsi]
    cmp al, 'e'
    jne error
    inc rsi
    dec byte[size]
    mov al, byte[rsi]
    cmp al, 'l'
    jne error
    inc rsi
    dec byte[size]
    mov al, byte[rsi]
    cmp al, 'e'
    jne error
    inc rsi
    dec byte[size]
    mov al, byte[rsi]
    cmp al, 't'
    jne error
    inc rsi
    dec byte[size]
    mov al, byte[rsi]
    cmp al, 'e'
    jne error
    inc rsi
    dec byte[size]
    jmp success1
success1:
    inc rsi
    dec byte[size]
    mov rdi, name2
top1:
    mov al, byte[rsi]
    mov [rdi], al
    inc rdi
    inc rsi
    dec byte[size]
    jnz top1
    mov rax, 87
    mov rdi, name2
    syscall
    jmp _start
COPY:
    print entercmd, cmdlen
    accept choice, chlen

    mov byte[size], al
    dec byte[size]
    mov al, byte[rsi]
nd2:
```

```
   cmp al, 'c'
   jne error
   inc rsi
   dec byte[size]
   mov al, byte[rsi]
   cmp al, 'o'
   jne error
   inc rsi
   dec byte[size]
   mov al, byte[rsi]
   cmp al, 'p'
   jne error
   inc rsi
   dec byte[size]
   mov al, byte[rsi]
   cmp al, 'y'
   jne error
   inc rsi
   dec byte[size]
   jmp success2
success2:
      inc rsi
   dec byte[size]
   mov rdi, name1
   mov rcx, 9
nsp:
   mov al, [rsi]
   mov [rdi], al
   inc edi
   inc rsi
   dec byte[size]
   loop nsp
   inc rsi
   dec byte[size]
      xor rdi, rdi
   mov rdi, name2
nnl:
   mov al, [rsi]
   mov [rdi], al
   inc rdi
   inc rsi
   dec byte[size]
   jnz nnl

   iomodule 2, name1, 2, 777
   mov qword[fdis1], rax

   iomodule 0, [fdis1], buffer, 200
   mov qword[cnt1], rax
   iomodule 2, name2, 2, 777
   mov qword[fdis2], rax
   iomodule 1, [fdis2], buffer, qword[cnt1]
   mov rax, 3
   mov rdi, name1
   syscall
   mov rax, 3
   mov rdi, name2
   syscall
jmp _start
   exit
```

OUTPUT:

```
swlab@swlab-H81-M1:~$ cd Desktop/
swlab@swlab-H81-M1:~/Desktop$ cd cmd/
swlab@swlab-H81-M1:~/Desktop/cmd$ nasm
-f elf64 cmd.asm
swlab@swlab-H81-M1:~/Desktop/cmd$ ld -o
cmd cmd.o
swlab@swlab-H81-M1:~/Desktop/cmd$ ./cmd
command menu
1.TYPE
2.COPY
3.DELETE
4.EXIT
1
Enter command:——: type file1.txt
I am prof Chaitanya Shimpi

command menu
1.TYPE
2.COPY
3.DELETE
4.EXIT
1
Enter command:——: type file2.txt
I am prof Chaitanya
command menu
1.TYPE
2.COPY
3.DELETE
4.EXIT
2
Enter command:——: copy file1.txt file2.txt
command menu
1.TYPE
2.COPY
3.DELETE
4.EXIT
3
Enter command:——: delete file1.txt
```

Microprocessor Experiment No. 13
Name: Swayam Shinde
Class: SE COMP-B
Roll No: 22CO114

Write x86 ALP to find the factorial of a given integer number on a command line by using recursion. Explicit stack manipulation is expected in the code.

```
section .data
msgFact: db &#39;Factorial is:&#39;,0xa
msgFactSize: equ $-msgFact
newLine: db 10
section .bss
fact: resb 8
num: resb 2
section .txt
global _start
_start:
pop rbx ;Remove number of arguments
pop rbx ;Remove the program name
pop rbx ;Remove the actual number whoes
factorial is to be calculated
(Address of number)
mov [num],rbx
;print number accepted from command line
mov rax,1
mov rdi,1
mov rsi,[num]
mov rdx,2
syscall

mov rsi,[num]
mov rcx,02
xor rbx,rbx

call aToH
mov rax,rbx
call factP

mov rcx,08
mov rdi,fact
xor bx,bx
mov ebx,eax
call hToA
mov rax,1
mov rdi,1
mov rsi,newLine
mov rdx,1
syscall
mov rax,1
mov rdi,1
mov rsi,fact
mov rdx,8
syscall
mov rax,1
mov rdi,1
mov rsi,newLine
mov rdx,1
syscall
mov rax,60
mov rdi,0
syscall
factP:
dec rbx
cmp rbx,01
je comeOut
cmp rbx,00
je comeOut
mul rbx
call factP

comeOut:
ret
aToH:
up1: rol bx,04
mov al,[rsi]
cmp al,39H
jbe A2
sub al,07H
A2: sub al,30H
add bl,al
inc rsi
loop up1
ret

hToA:
   d:  rol ebx,4
mov ax,bx
and ax,0fH
cmp ax,09H
jbe ii
add ax,07H
```

```
ii: add ax,30H
mov [rdi],ax
inc rdi
loop d
ret
```

OUTPUT:

```
;nasm -f elf64 ass9_rec.asm
; ld -o ass9_rec ass9_rec.o
;./ass9_rec 02
;02
;00000002
```