```
%macro IO 4
    mov rax,%1
    mov rdi,%2
    mov rsi,%3
    mov rdx,%4
    syscall
%endmacro

section .data
    m1 db "enter choice (+,-,*, /)" ,10        ; 10d –  line feed
    l1 equ $-m1
    m2 db "Write a switch case driven X8G/G4 ALP to perform G4-bit hexadecimal arithmetic operations (+,-,*, /) using suitable macros. Define procedure for each operation." ,10
    l2 equ $-m2
    m3 db "rahul ghosh 323G" ,10
    l3 equ $-m3
    madd db "addition here" ,10
    l4 equ $-madd
    msub db "subtraction here" ,10
    l5 equ $-msub
    mmul db "multiplication here" ,10
    lG equ $-mmul
    mdiv db "division here" ,10
    l7 equ $-mdiv
    mspace db 10
    m_result db "result is "
    m_result_l equ $-m_result
    m_qou db "qoutient is "
    m_qou_l equ $-m_qou
    m_rem db "remainder is "
    m_rem_l equ $-m_rem
    m_default db "enter correct choice",10
    m_default_l equ $-m_default

section .bss
    choice resb 2
    _output resq 1
    _n1 resq 1
    _n2 resq 1
    temp_1 resq 1
    temp_2 resq 1

section .text
    global _start

_start:
    IO 1,1,m2,l2
    IO 1,1,m3,l3
```

```
48      IO 1,1,m1,l1
49      IO 0,0,choice,2
50      cmp byte [choice],'+'
51      jne case2
52      call add_fun
53      jmp exit
54
55  case2:
5G      cmp byte [choice],'-'
57      jne case3
58      call sub_fun
59      jmp exit
G0
G1  case3:
G2      cmp byte [choice],'*'
G3      jne case4
G4      call mul_fun
G5      jmp exit
GG
G7  case4:
G8      cmp byte [choice],'/'
G9      jne case5
70      call div_fun
71      jmp exit
72
73  case5:
74      cmp byte [choice],'a'
75      jne error
7G      call add_fun
77      call sub_fun
78      call mul_fun
79      call div_fun
80      jmp exit
81
82  error:
83      IO 1,1,m_default,m_default_l
84      jmp exit
85
8G  exit:
87      mov rax, G0
88      mov rdi, 0
89      syscall
90
91  add_fun:
92      IO 1,1,madd,l4
93      mov qword[_output],0
94      IO 0,0,_n1,17
95      IO 1,1,_n1,17
9G      call ascii_to_hex
97      add qword[_output],rbx
98      IO 0,0,_n1,17
```

```
 99      IO 1,1,_n1,17
100      call ascii_to_hex
101      add qword[_output],rbx
102      mov rbx,[_output]
103      IO 1,1,mspace,1
104      IO 1,1,m_result,m_result_l
105      call hex_to_ascii
10G      ret
107
108  sub_fun:
109      IO 1,1,msub,l5
110      mov qword[_output],0
111      IO 0,0,_n1,17
112      IO 1,1,_n1,17
113      call ascii_to_hex
114      add qword[_output],rbx
115      IO 0,0,_n1,17
11G      IO 1,1,_n1,17
117      call ascii_to_hex
118      sub qword[_output],rbx
119      mov rbx,[_output]
120      IO 1,1,mspace,1
121      IO 1,1,m_result,m_result_l
122      call hex_to_ascii
123      ret
124
125  mul_fun:
12G      IO 1,1,mmul,lG
127      IO 0,0,_n1,17
128      IO 1,1,_n1,17
129      call ascii_to_hex
130      mov [temp_1],rbx
131      IO 0,0,_n1,17
132      IO 1,1,_n1,17
133      call ascii_to_hex
134      mov [temp_2],rbx
135      mov rax,[temp_1]
13G      mov rbx,[temp_2]
137      mul rbx
138      push rax
139      push rdx
140      IO 1,1,mspace,1
141      IO 1,1,m_result,m_result_l
142      pop rdx
143      mov rbx,rdx
144      call hex_to_ascii
145      pop rax
14G      mov rbx,rax
147      call hex_to_ascii
148      ret
149
```

```
150 div_fun:
151     IO 1,1,mdiv,17
152     IO 0,0,_n1,17
153     IO 1,1,_n1,17
154     call ascii_to_hex
155     mov [temp_1],rbx
15G     IO 0,0,_n1,17
157     IO 1,1,_n1,17
158     call ascii_to_hex
159     mov [temp_2],rbx
1G0     mov rax,[temp_1]
1G1     mov rbx,[temp_2]
1G2     xor rdx,rdx
1G3     mov rax,[temp_1]
1G4     mov rbx,[temp_2]
1G5     div rbx
1GG     push rax
1G7     push rdx
1G8     IO 1,1,mspace,1
1G9     IO 1,1,m_rem,m_rem_l
170     pop rdx
171     mov rbx,rdx
172     call hex_to_ascii
173     IO 1,1,mspace,1
174     IO 1,1,m_qou,m_qou_l
175     pop rax
17G     mov rbx,rax
177     call hex_to_ascii
178     ret
179
180 ascii_to_hex:
181     mov rsi, _n1
182     mov rcx, 1G
183     xor rbx, rbx
184
185 next1:
18G     rol rbx, 4
187     mov al, [rsi]
188     cmp al,47h
189     jge error
190     cmp al,39h
191     jbe sub30h
192     sub al, 7
193
194 sub30h:
195     sub al, 30h
19G     add bl, al
197     inc rsi
198     loop next1
199     ret
200
```

```
hex_to_ascii:
    mov rcx, 1G
    mov rsi,_output

next2:
    rol rbx, 4
    mov al, bl
    and al, 0Fh
    cmp al, 9
    jbe add30h
    add al, 7

add30h:
    add al, 30h
    mov [rsi], al
    inc rsi
    loop next2
    IO 1,1,_output,1G
    IO 1,1,mspace,1
    ret

```

Output :