MOV eax, 0F0F0F0F0h;
MOV edx, 87654321h;
NOT ax;
INC dx;
AND edx, eax;

NOT ax; ->0F0F00F0Fh;
INC dx; -> 87654322h;
**AND : -> 80600302h;**

---

Zmienna al: 0a5h (h – hex)
Do postaci: 5ah (h – hex)
Korzystamy z funkcji XOR al, 255;
**0101 1010 = 5a**

---

SHL – przesunięcie logiczne w lewo:
Przed: 0000 1010 0101 <- 0
**Po 0001 0100 1010**

---

ROR – obrót bitów w prawo (tu o 1 bit)
Przed: 0000 1010 0101 ->
Po **1000 0101 0010**

---

Podaj instrukcję MMX pakującą z nasyceniem słowa ze znakiem do bajtów: **PACKSSWB**

---

Napisz program ustawiający bit 8, zerujący bit 4
I zmieniający bit 2 w zmiennej całkowitej a.

```
__asm{
    MOV eax, 01111111b;
    AND  eax, 11110111b;
    OR    eax, 10000000b;
    XOR  eax, 10b;
    }
```

---

1. Napisz program obliczający ln x.
Wskaźnik do x znajduje się w eax, wynik pozostaw w rejestrze st(0):

```
fld [eax] ;          x
;                    loge x = log2 x/log2 e
fld1 ;               1
fyl2x ;              log2 x
fldl2e ;             log2 e, log2 x
fdivp st(1), st ;    log2 x/log2 e
```

---

1. Napisz przy użyciu instrukcji łańcuchowych program przesyłający 555 bajtów z tablicy tab 1 do tablicy tab 2

```
mov eax, 555
mov esi, tab1
mov edi, tab2
cld
rep movsb
```

---

Napisz program umieszczajacy w eax zaokrągloną średnią z eax i edx

```
add eax, edx ;
mov temp, eax ;

;czas na koprocesor

fld temp ;
fld1 ;
fld1 ;
faddp ;
fdivp ;
frndint ;
fist wynik

lub

add eax, edx;
sar  eax, 2;
jnc koniec;
inc eax;
koniec:
```

---

Proszę napisać fragment programu w asemblerze realizujący to samo zadanie co fragment kody w C++

```
int mniejsze( int a, int b)
{
int x=0;
if(a<b)
x=a(b-a);
return x;
}
int funkcja(int a, int b) {
  int wynik;
  __asm {
    mov eax, 0 ;wynik = 0
    mov ebx, a ;ebx = a
    mov ecx, b ;ecx = b
    cmp ebx, ecx
    jge koniec ;jesli a>=b -> idz do konca, bo
nie ma co robic
    sub ecx, ebx ;ecx = ecx - ebx (b-a)
    imul ecx, ebx ;ecx = ecx*ebx (roznica*a)
    mov eax, ecx ;eax = wynik
  koniec:
    mov wynik, eax
  }
  return wynik;
}
```

Uzupełnij poniższy program tak aby odnajdywał najmniejszą wartość z tablicy tab

```
__asm{
   mov esi , tab;
   mov ecx , N;
   mov eax {esi + 4*ecx - 4  };
   dec ecx;
Label 1:
   cmp eax , {esi + 4*ecx – 4 };
    jle  lab2:
    mov eax , {esi + 4*ecx - 4 };
lab2:
    Loop label1;
}
```

---

Program który sprawdza ile z 345 słów z tab1 jest różnych od 345. Wynik umieść w eax:

```
MOV esi, tab1;
MOV exc, 345;
XOR eax, eax;
MOV ebx, ecx;
petla:
CMP ebx, [esi+4 * ecx-4];
JNE skok;
INC eax;
skok:
LOOP petla;
MOV w, eax;
```

---

Sinus: float alfa, float y; float b = 180;

```
cin >> alfa;
//
FLD alfa;
FLDPI;
FMUL;
FLD b;
FDIV;
FSIN;
FSTP y;
```

---

Napisz program obliczający log_a x.
Wskaźniki do a oraz x znajdują się w eax i edx, wynik pozostaw w rejestrze st(0).

```
FLD 1;
FLD x;
FYL2X;
FLD1;
FLD x;
FYL2X;
FDIV;
```

---

DZIELENIE jakies:

```
SUB eax, eax;  XOR edx, edx;
MOV eax, dzielna;  MOV ebx, dzielnik;
DIV ebx;   MOV wynik, eax;
```

Przekątna prostopadłościanu:

```
 int n = 10;
 int *tab1 = new int[10];
 int *tab2 = new int[10];
 __asm{
        MOV ecx, n;
        MOV esi, tab1;
        MOV edi, tab2;
        FLD qword ptr[esi];
        FLD qword ptr[edi];
        FMUL;
        DEC ecx;
SKOK:
        ADD esi, 8;
        ADD edi, 8;
        FLD qword ptr[esi];
        FMUL qword ptr[edi];
        FADD;
        DEC ecx;
        JNZ skok;
```

---

Napisz program zamieniający kolejność bitów w zmiennej całkowitej a:

```
        MOV eax, a;
        MOV ecx, 31;
        MOV ebx, 31;
        XOR edx, edx;
 petla:
        PUSH ebx;
        BT eax, ecx;
        JC ustaw;
        JMP koniec;
 ustaw:
        SUB ebx, ecx;
        BTS edx, ebx;
        JMP koniec;
koniec:
        POP ebx;
        DEC ecx;
        CMP ecx, 0;
        JGE petla;
        MOV wynik, edx;
```

---

Obliczyć y = e^x (coś w tym stylu)

```
FLD    x;
FLDe;   // e, x
FYL2X; // x * log2(e)
FLD st; // x * log2(e); x * log2(e)
FRNDINT; // round …
FSUB st(1), st;
FXCH;
F2XM1;
FLD1;
FADD;
FSCALE;
FSTP koniec; (resb)
```