|  |  |
| --- | --- |
| Data segment  ; repr fara semn  (a+b) – (11b – c)  A byte  B word  C double word  A db 10  B dw 20  C dd 10b  ; al | Code segment  ; (a+b)  Mov al, [A]  MOV AH, 0 ; AX -< [A]  ADD AX, [B] ; AX=AX+B  ; (11b – c)  ;SUB DEST, SOURCE ; DEST=DEST-SOURCE  MOV EBX, 11B  ;MOV ECX, [C]  ;SUB EBX, ECX  SUB EBX, [C] ; EBX= (11b – c)  MOV ECX, 0  MOV CX, AX ; ECX = (a+b)  SUB ECX, EBX ; ECX = ECX-EBX = (a+b) – (11b – c) |
| Mul byte\*byte  A db 10  Constanta 3  10 \* 3  Mul word\*word  B dw 5  C dw 6  B\*c  Mul doubleword\*doubleword  M dd 5  N dd 8  Byte\*word   * Word\*word   A db 2  B dw 4  (a-2)\*b  4/3  Byte/byte  b->word  r/s, r s byte  word/word  g/h, g, h word  byte/word  f/e f byte, e word  quadword/doubleword  j/i  j dq, i dd  ; reprez fara semn  a / (2- b) + (c+1)\*(d-1)  a doubleword  b byte  c word  d byte  a dd 40  b db 3  c dw 5  d db 9  auxx dw 0  aux dd 0 | Mov al, [a]  Mov bl, 3  Mul bl ; bl \*al = ax  ;MUL 3 NU! Op explicit nu poate fi constanta  V 2  Mov al, 3  Mul byte[a] ; ax = a\*al = a\*3  Mov ax, [b]  Mov bx, [c]  Mul bx ; bx\*ax = DX:AX  Mov eax, [m]  Mov ebx, [n]  Mul ebx ; eax\*ebx=edx:eax  ;v2  Mov eax, [m]  Mul dword [n] ; edx:eax = m\*n  Mov al, [a]  Sub al, 2 ; al=a-2  Mov ah, 0 ; ax= [a]  Mult word[b] ; DX:AX = ax\*b  Mov ax, 4  Mov bl, 3  Div bl ; al catul imp lui 4 la 3 si in ah avem restul impartirii lui 4/3  Mov al, [r]  Mov ah, 0 ;ax avem [R ]  Div byte [s] ; al cat si ah rest  Mov ax, [g]  Mov dx, 0 ;dx:ax = avem g pe doubleword  Div word[h] ; ax catul impartirii si dx restul impartirii  Mov al, [f]  Mov ah, 0  Mov DX, 0 ;dx:ax avem f  Div word[e] ;ax cat, dx rest  J in combinatia de registrii edx, eax  Mov eax, dword [j+0]  Mov edx, dword [j+4] ;edx:eax este j  Div dword[i] ; eax- cat, edx avem rest  Byte – 8 biti al, ah, bl,bh, cl,ch,dl,dh  Word -16 biti ax, bx, cx, dx  Doubleowrd – 32 biti eax, edx, ebx, ecx sau combinatia de reg: DX:AX, Cx:Bx  Quadword – 64 biti edx:eax ; ecx:ebx  ; (2-b)  ;~~Mov bl, 2~~  ~~;Sub bl, [b] ; bl=b-2~~  Neg byte[b]  Add byte[b], 2 ; -b+2  Mov bl, [b]  Mov bh, 0 ; bx =[b]  ; a in dx:ax pt impartire  Mov ax, word [a+0]  Mov dx, word[a+2] ; dx:ax = [a]  Div bx ; dx:ax la bx, in ax cat (si in dx rest) ax= cat de la a / (2- b)  ; salvare pe ax in dx  Mov dx, ax  Mov [auxx],dx  ;(c+1)  Inc word [c] ; c=c+1  Mov bx, [c] ; bx= (c+1)  ; (d-1)  Dec byte[d] ; d=d-1  Mov cl, [d] ; cl =d-1  Mov ch, 0 ; cx = d-1  Mov ax, bx  Mul cx ; cx\* ax = dx:ax = a / (2- b) + (c+1)\*(d-1)  Mov bx, [auxx]  Mov cx, 0 ; cx:bx =[auxx] extins  ; adunare cx:bx+  Dx:ax  Add ax, bx  Adc dx, cx ; dx+cx+transportul de la op anterioara CF =se salveaza val 1 cand exista transport  ; rez final in dx:ax  ;dx:ax ->aux  ; little-endian  X db 12h  Y dw 3456h  Z dd 78901234h  T dq 1122334455667788h  Edx:EAX  X, y, z, t in memorie cf little-endian  12 56 34 34 12 90 78 88 77 66 55 44 33 22 11 X+0 y+0 y+1 z+0 z+1 z+2 z+3 t+0 t+1 t+2 t+4 t+7  T in edx:eax  Mov eax, dword [t+0]  Mov edx, dword [t+4]  dx:ax ->aux  78\_\_ 32 67 \_\_ 45\_  Aux+0 aux+1 aux+2 aux+3  Dx:ax - >aux (dx = 4567h, ax = 3278h)  Mov word [aux+ 0 ], ax  Mov word [aux + 2], dx  Aux -> dx:ax  Mov ax, word [aux+0]  Mov dx, word [aux+2] |