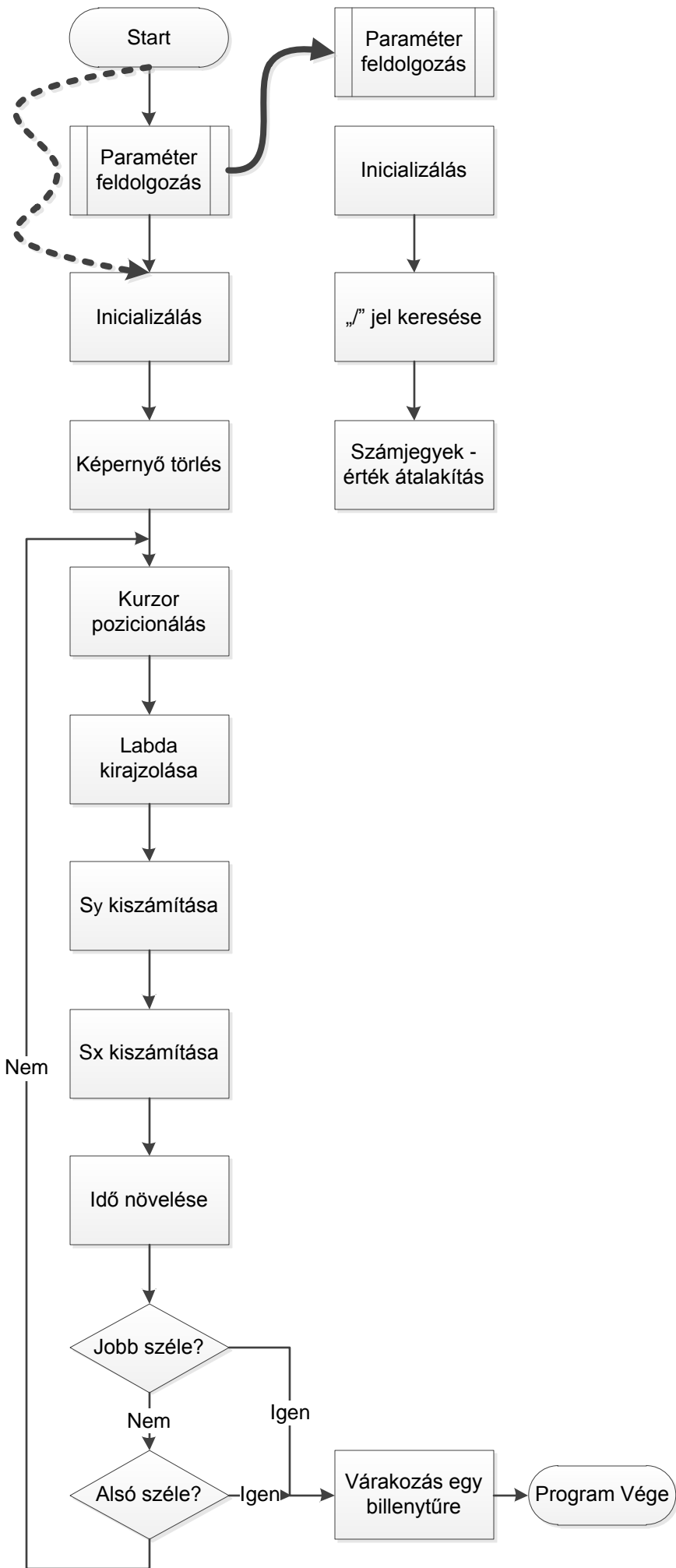
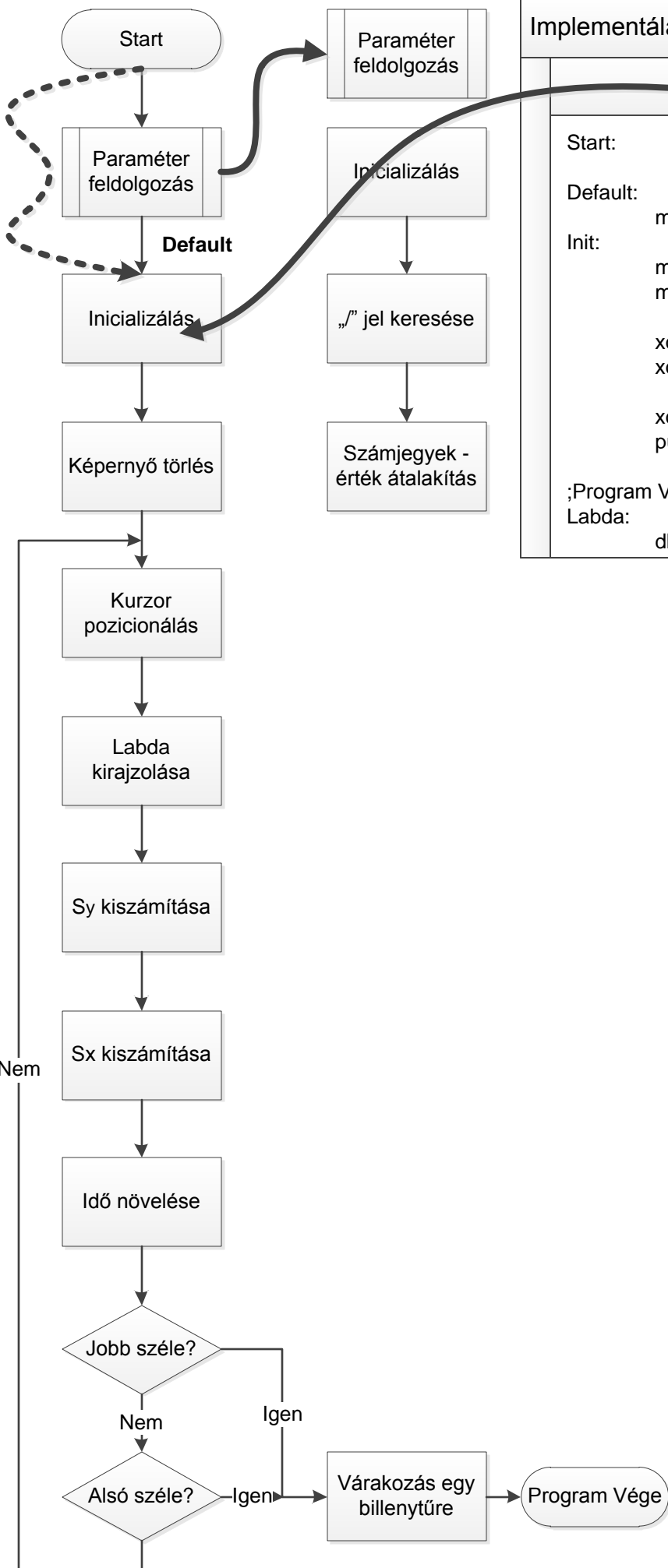


Fealadat4: param.asm

Feladat meghatározása	Implementálás	Implementálás
<p>A program célja bemutatni a parancssoros paraméterátadást. A hallgatók megismerik a PSP felépítését, szerepét, valamint annak elérését. A program az előző programhoz hasonlóan egy golyó leesését szimulálja. Paraméterként megadható a golyó vízszintes irányú kezdősebessége. Ekkor a test egy parabola pályán mozog, amit a program kirajzol. Paraméter nélkül indítva a programot, az egy alapértelmezett (10m/s) kezdősebességű testet szimulál. A számítás alapja:</p> $s_y = \frac{1}{2} * g * t^2$ $s_x = v_x * t$ <p>Program bemutatása:</p> <ul style="list-style-type: none"> A programot először a bemenő paramétereket feldolgozó rész nélkül kell megírni. Ekkor a program az alapértelmezett 10m/s kezdősebességgel induló test pályáját rajzolja ki. A működő programot ki kell egészíteni a paraméter feldolgozó résszel. Az így megírt program szigorúan „/11” formátumban fogadja a paramétereket. Önálló gyakorlásként ajánlható a paraméter feldolgozás hibakezelésének megírása. Például a program ne számoljon a „/w2” paraméterrel. 	<pre>Code Segment assume CS:Code, DS:Data, SS:Stack Start: mov di, 82h Keres: mov cx, 10 mov dl, [di] cmp dl, "/" jz ParamKezdet inc di loop Keres jmp Default ParamKezdet: inc di mov bl, [di] sub bl, 48 inc di mov bh, [di] sub bh, 48 mov ax, 10 mul bl add al, bh mov cx, ax jmp Init Default: mov cx, 10 Init: mov ax, Code mov ds, ax xor di, di xor si, si xor dx, dx push dx Torles: mov ax, 03h int 10h Rajz: mov bx, di mov dh, bl mov bx, si mov dl, bl xor bh, bh mov ah, 02h int 10h mov dx, offset Labda mov ah, 09h int 21h</pre>	<pre>pop ax push ax mov bl, al mul bl shr ax, 1 mov di, ax pop ax inc ax push ax dec ax mov bl, cl mul bl mov si, ax cmp si, 80 jnc Var cmp di, 25 jnc Var jmp Rajz Var: xor ax, ax int 16h Program_Vege: pop cx mov ax, 4c00h int 21h Labda: db "O\$" Code Ends Data Segment Data Ends Stack Segment Stack Ends End Start</pre>





Implementálás

Inicializálás

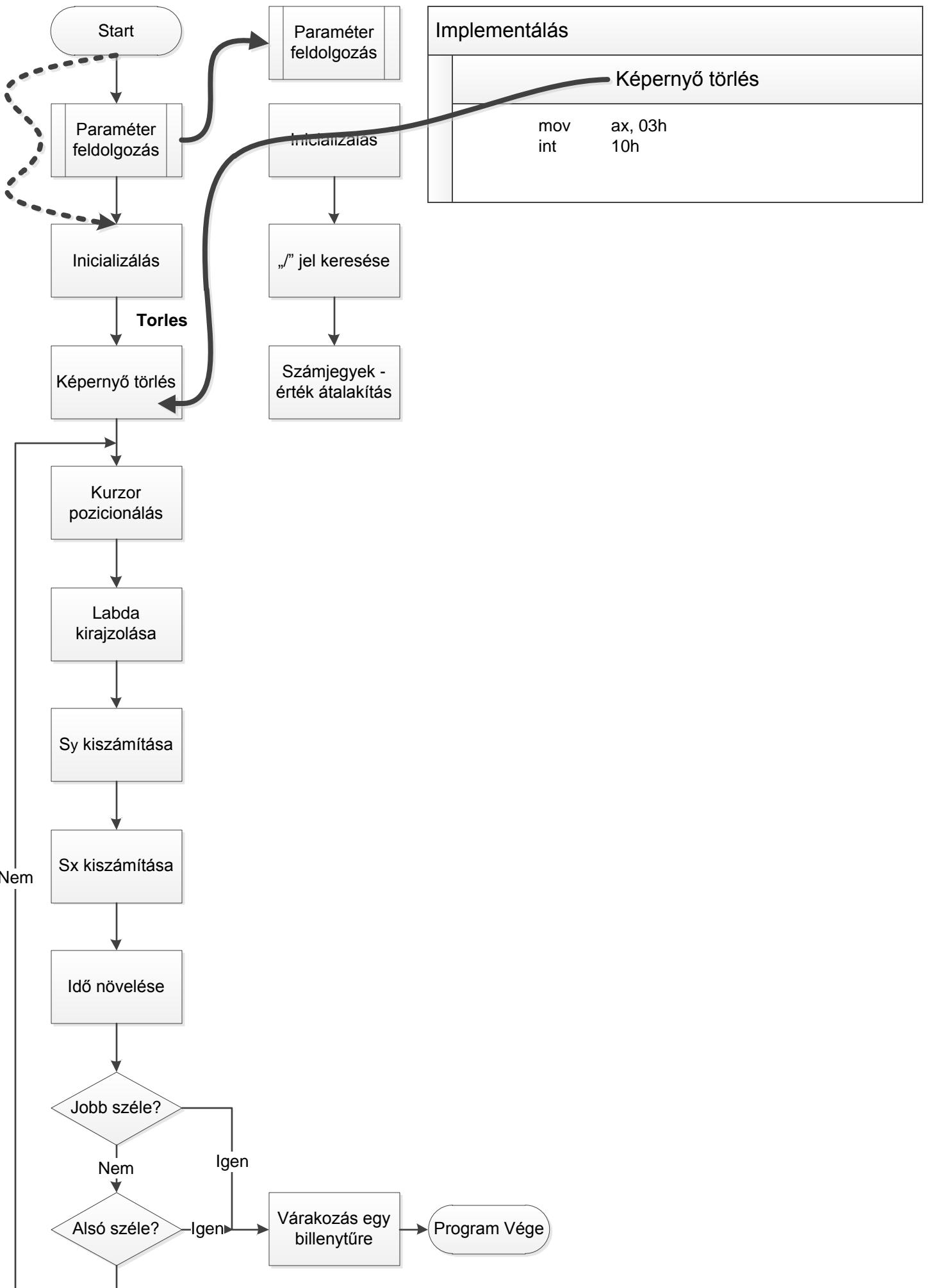
```

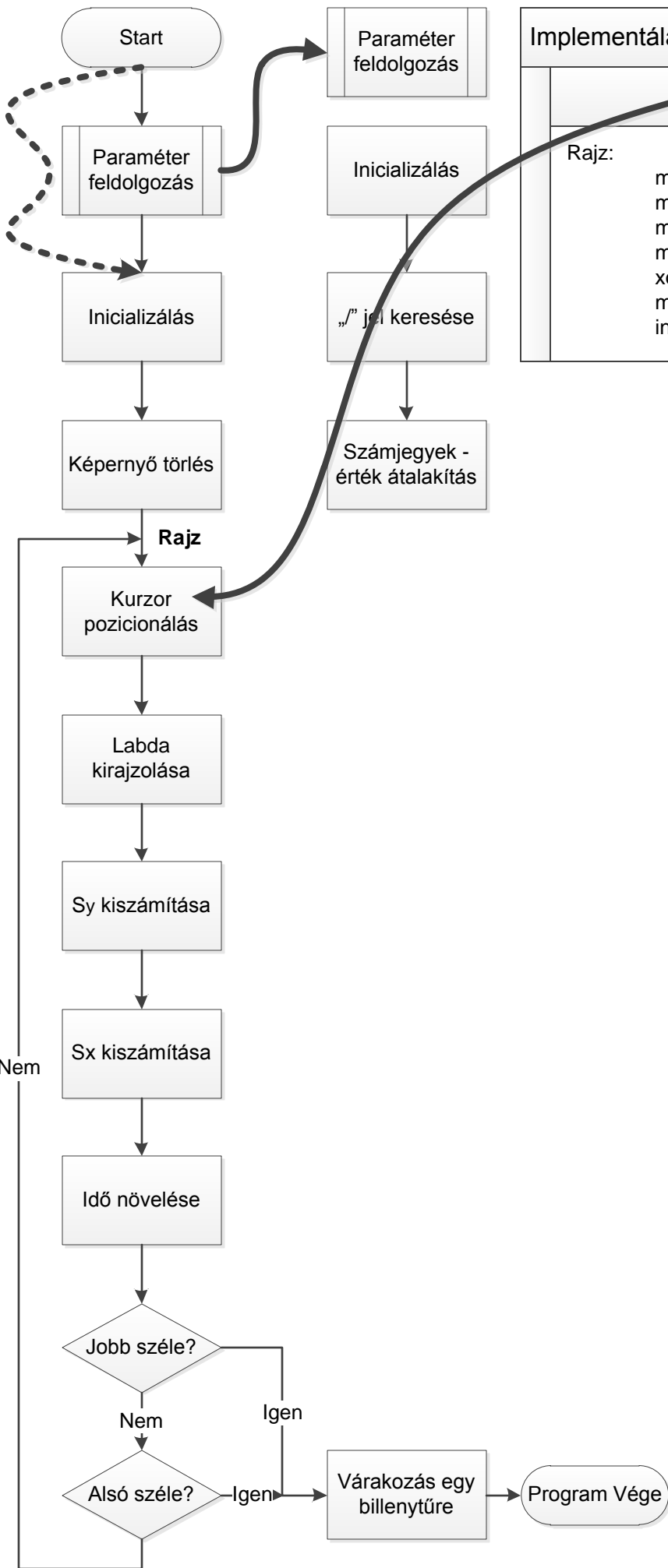
Start:
Default:
Init:
  mov     cx, 10      ;Alapértelmezett v0
  mov     ax, Code
  mov     ds, ax

  xor     di, di      ;golyó sor pozíciója
  xor     si, si      ;golyó oszlop pozíciója

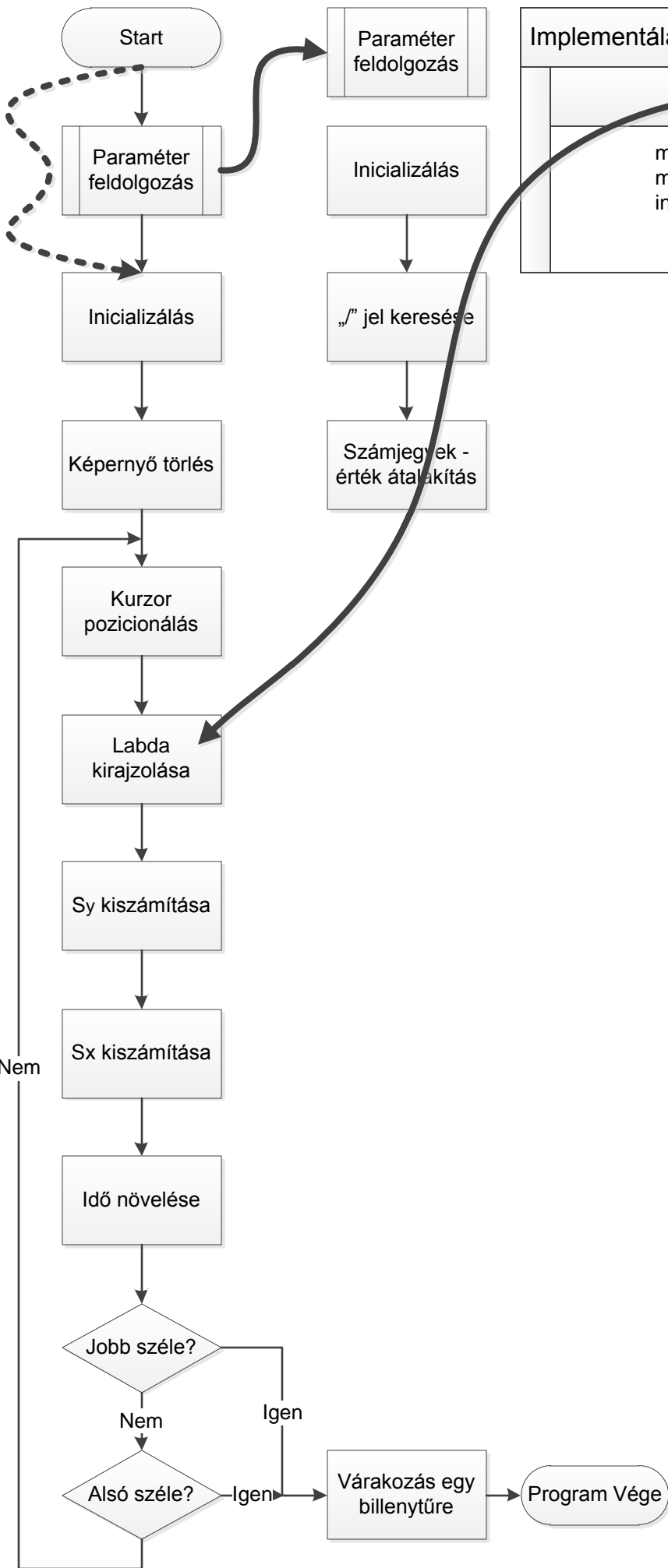
  xor     dx, dx
  push   dx          ;verembe az idő (most 0)

;Program Vége után, de még a Code szegmensbe
Labda:
  db     "0$"
  
```





Implementálás			
Kurzor pozicionálás			
Rajz:	mov	bx, di	;sor koordináta
	mov	dh, bl	;di alsó bájta tartalmazza
	mov	bx, si	;oszlop koordináta
	mov	dl, bl	;si alsó bájta tartalmazza
	xor	bh, bh	
	mov	ah, 02h	
	int	10h	



Implementálás	
Labda kirajzolása	
mov	dx, offset Labda
mov	ah, 09h
int	21h

Paraméter feldolgozás

Inicializálás

„/” jel keresése

Számjegyek -
érték átalakítás

Start

Paraméter feldolgozás

Inicializálás

Képernyő törlés

Kurzor pozicionálás

Labda kirajzolása

Sy kiszámítása

Sx kiszámítása

Idő növelése

Jobb széle?

Alsó széle?

Várakozás egy
billentyűre

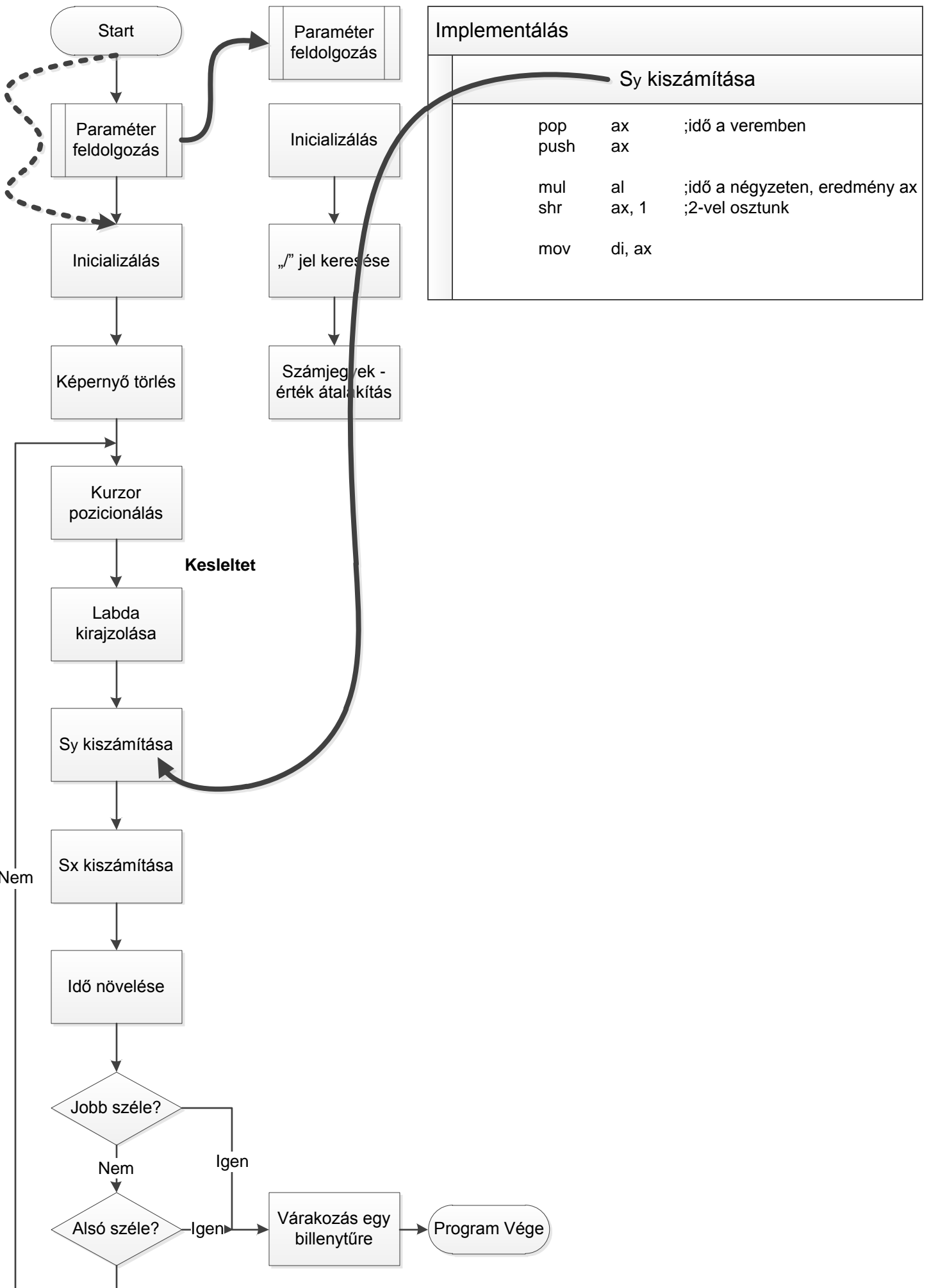
Program Vége

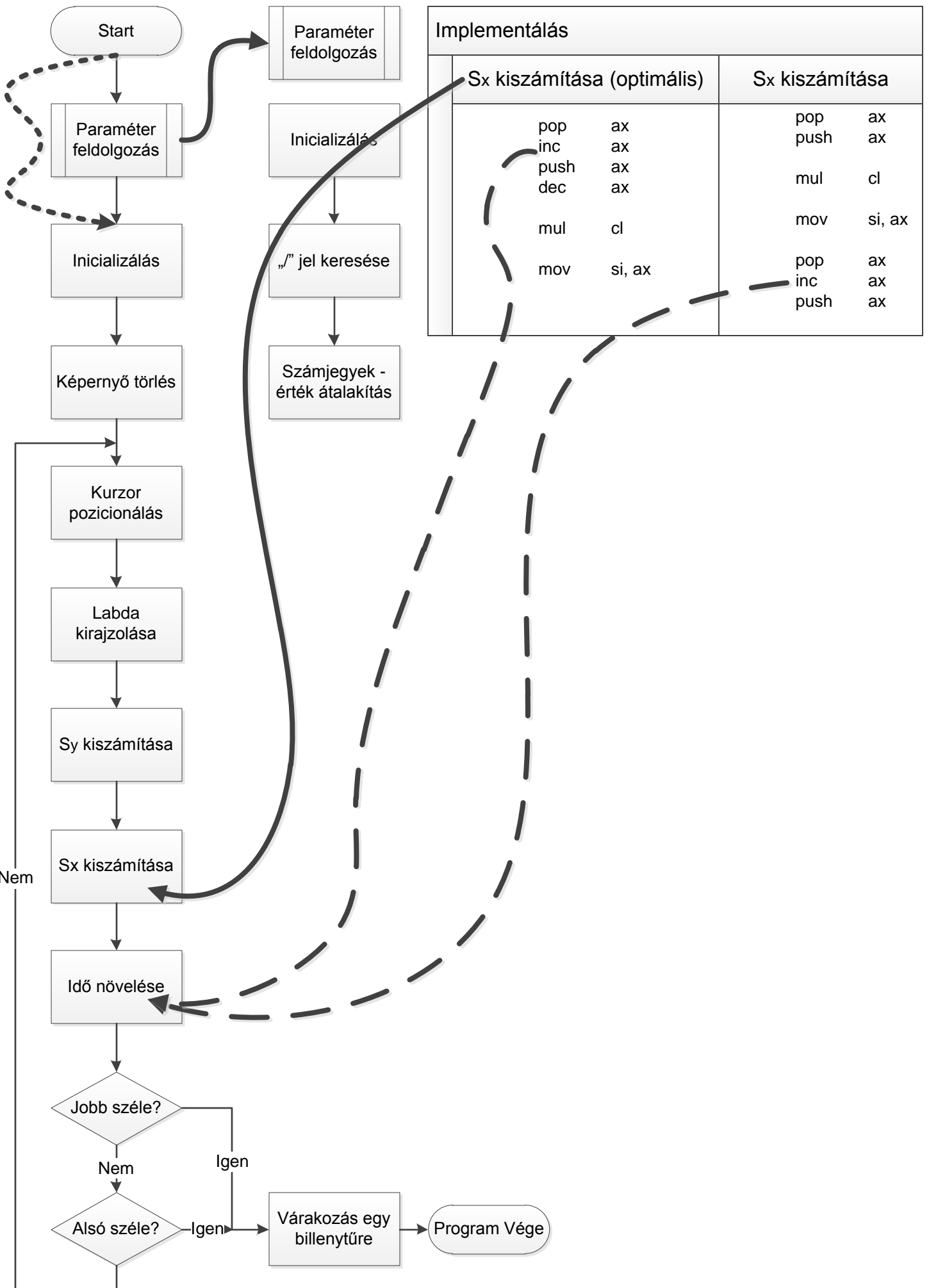
Nem

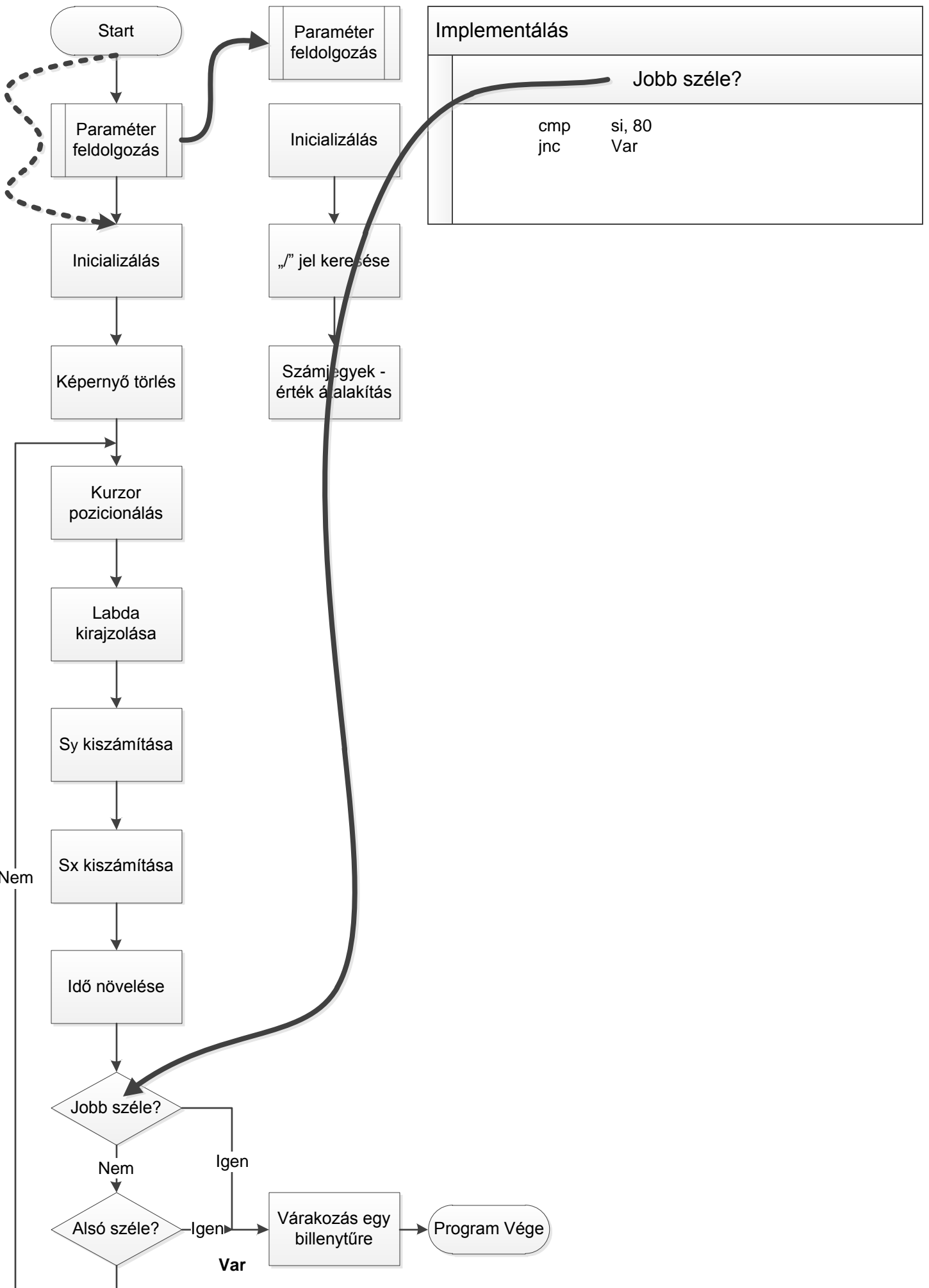
Igen

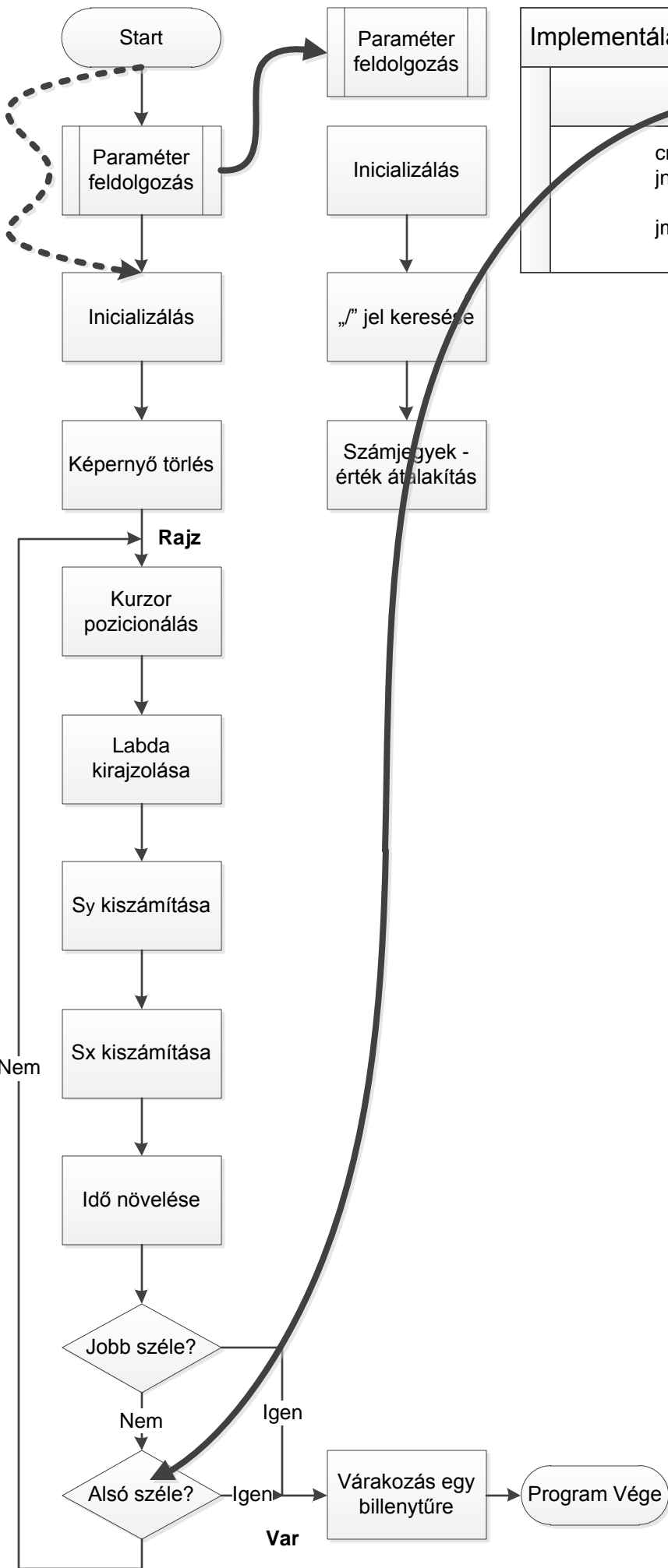
Nem

Igen



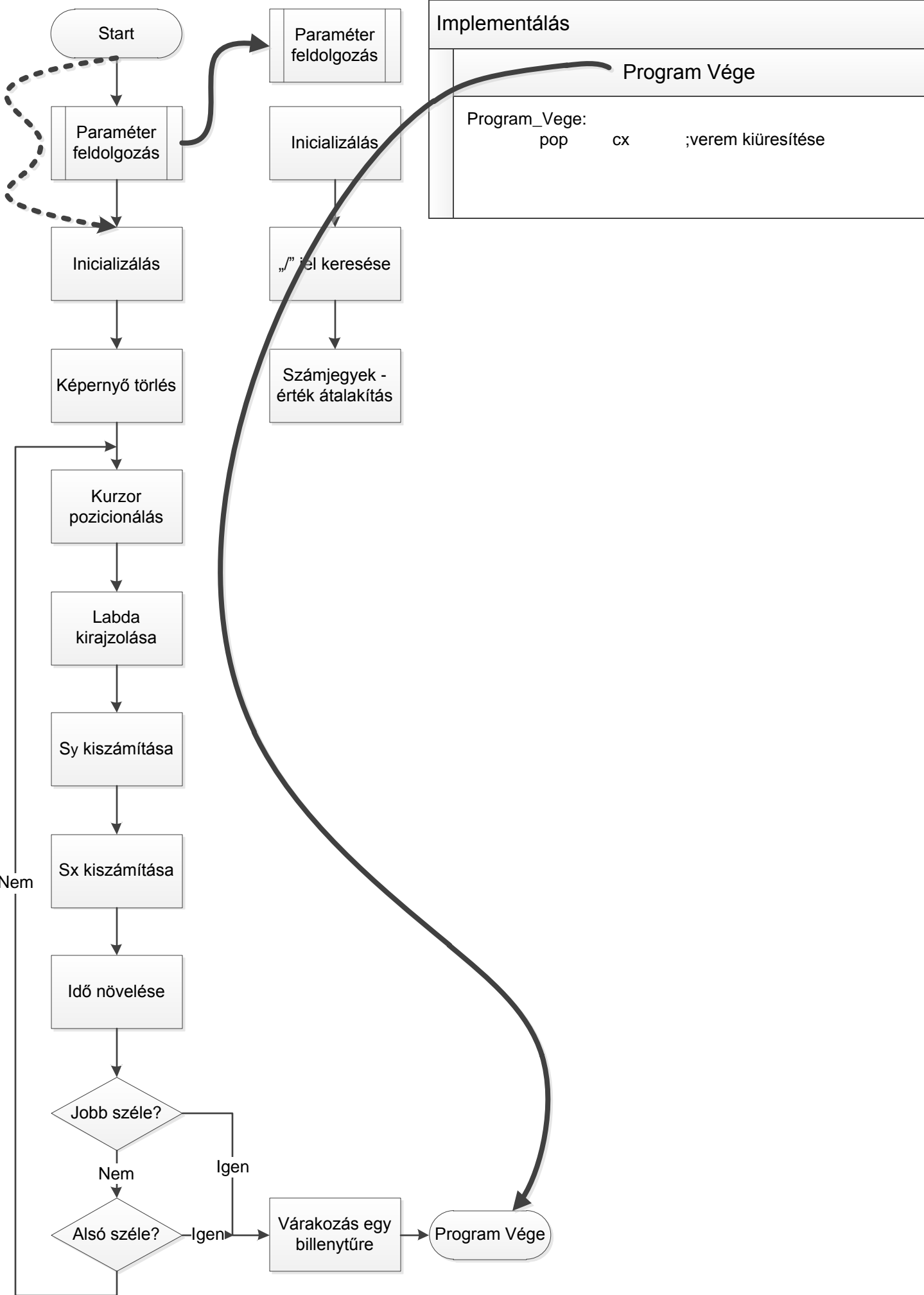


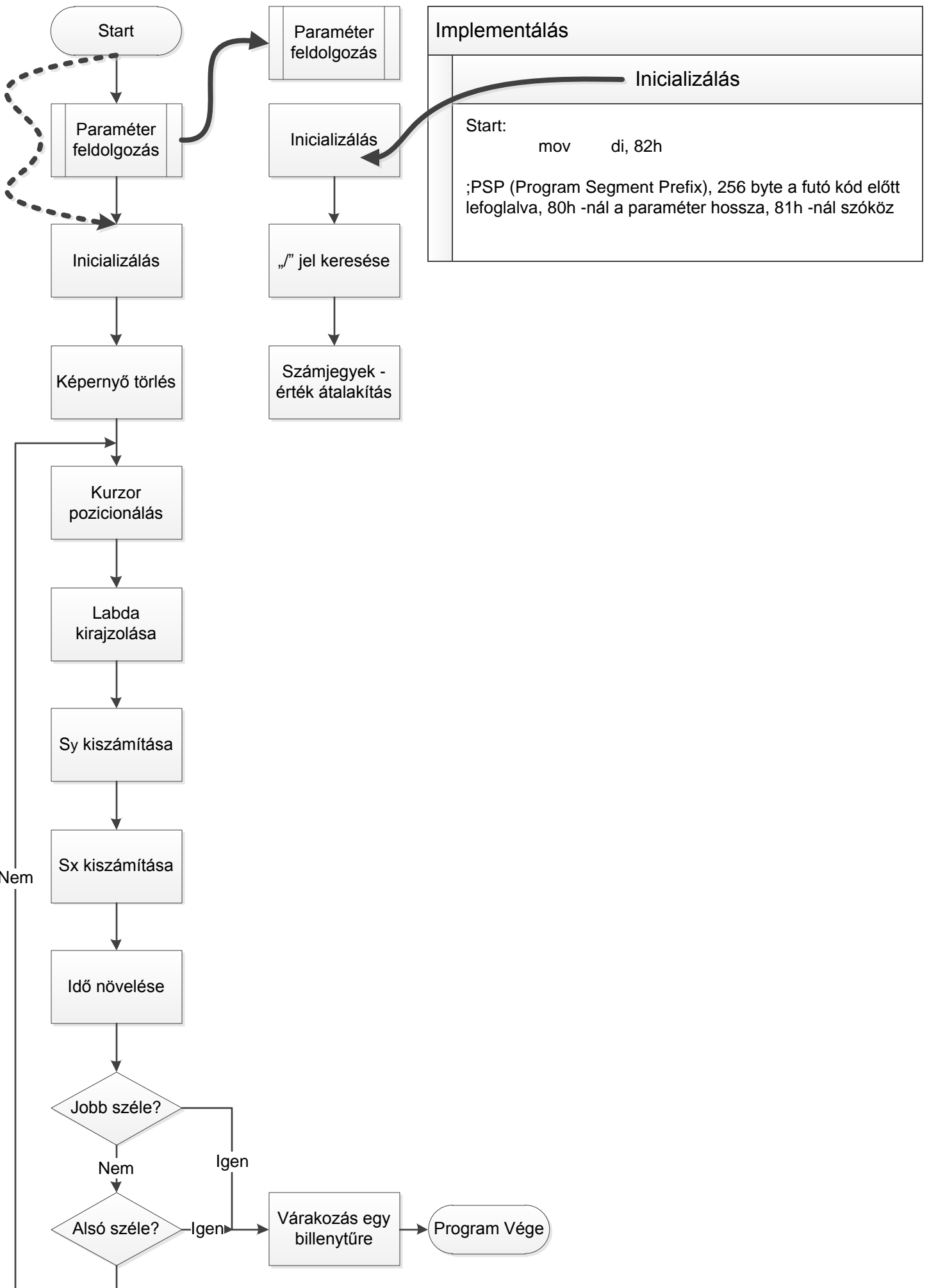


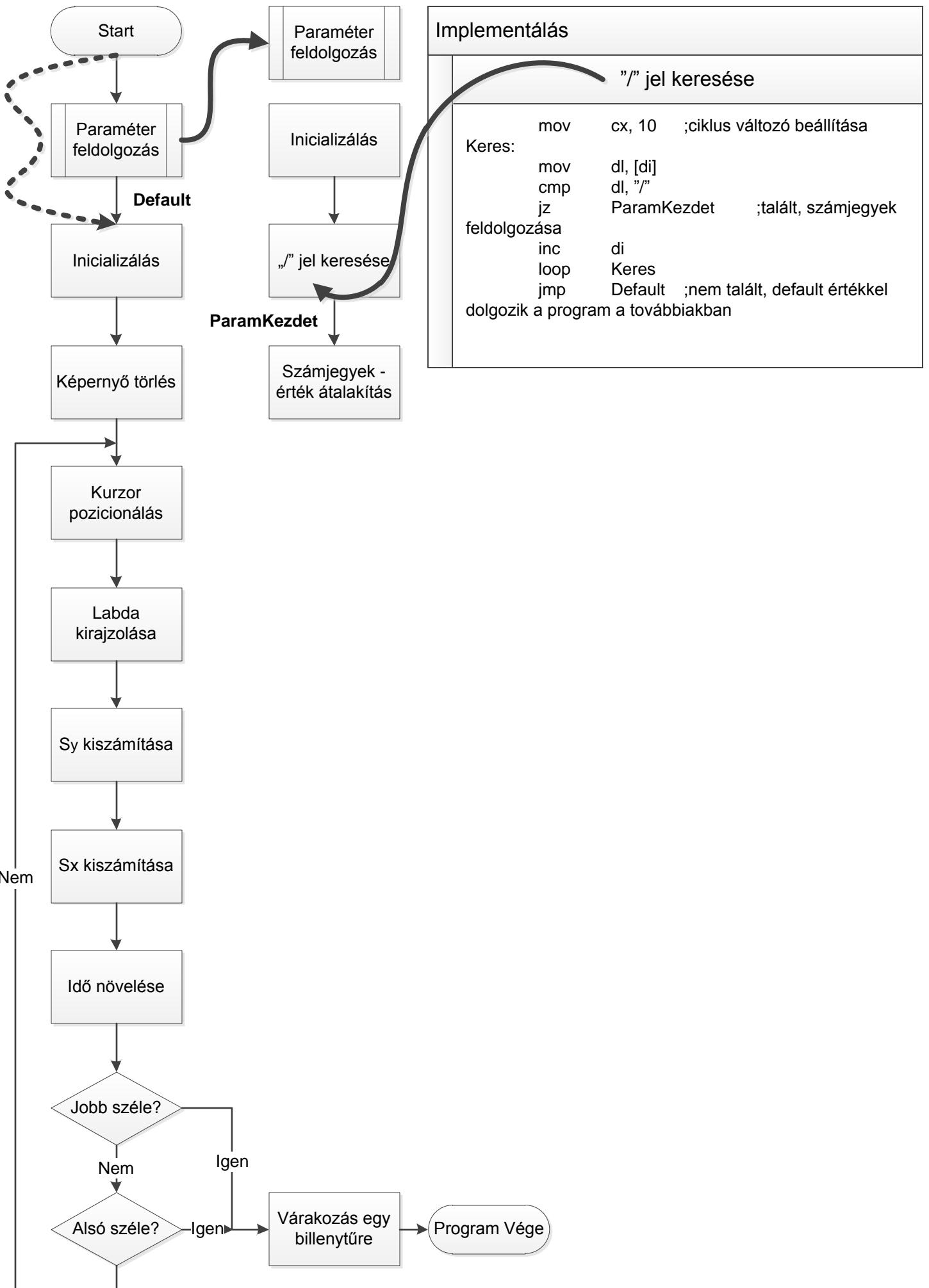


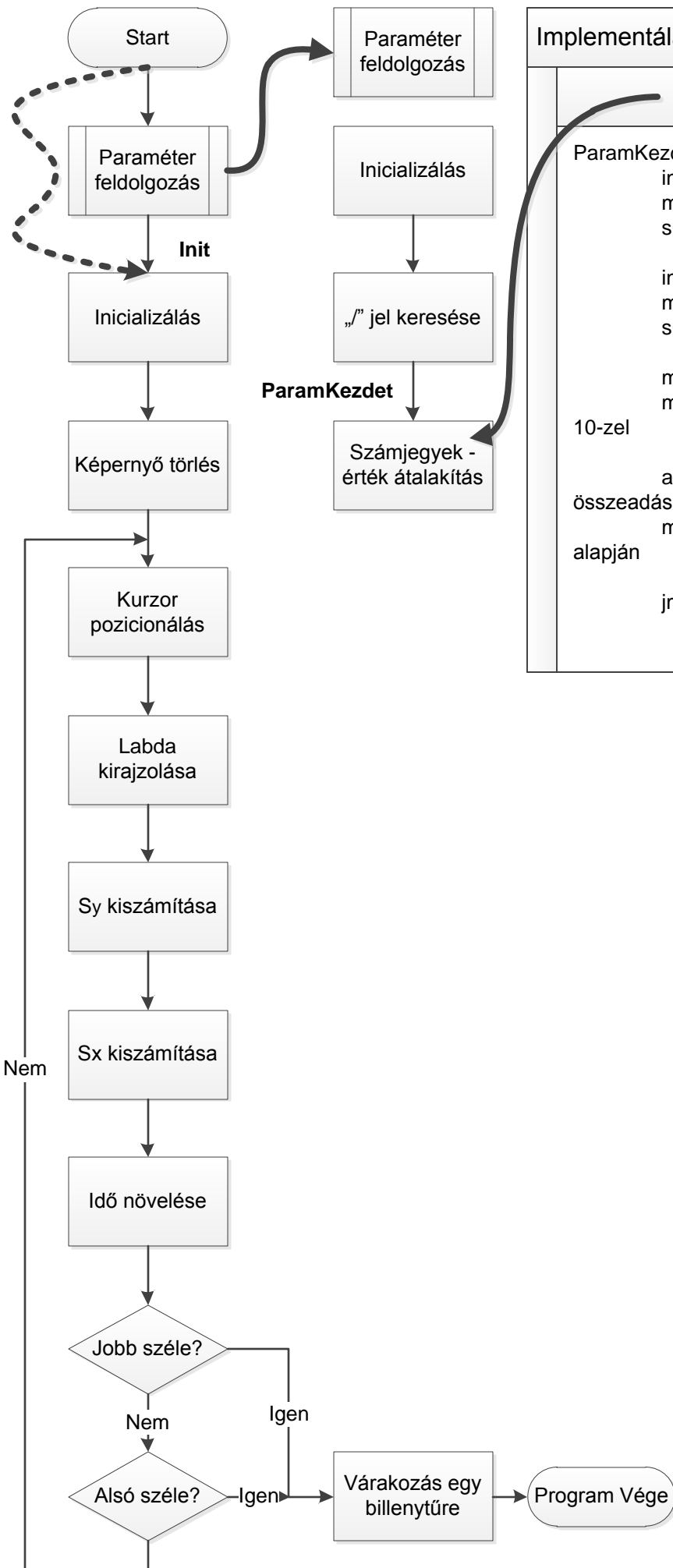
Implementálás

Alsó széle?	
cmp	di, 25
jnc	Var
jmp	Rajz









Implementálás

Számjegyek - érték átalakítás

```

ParamKezdet:
    inc    di
    mov    bl, [di] ;első számjegy, tízesek
    sub    bl, 48

    inc    di
    mov    bh, [di] ;második számjegy, egyesek
    sub    bh, 48

    mov    ax, 10
    mul    bl ;tízes számjegy megszorozása
    10-zel

    add    al, bh ;tízesek és egyesek
    összeadása
    mov    cx, ax ;v0 beállítása a paraméter
    alapján

    jmp    Init
  
```