TVC függőleges hardware scroll elmélet

Hosszú lesz, szóval hozz egy italt! :)

A **TVC**-s hardware scroll egyszerű dolog, hiszen csak a **CRTC 6845**-el a képernyő kezdőcímét kell folyamatosan **64**-el növelni vagy csökkenteni és szépen megy is a scroll fel vagy le. Egészen addig van ez így, amíg nem akarja az ember, hogy az ne "röcögve" scrollozzon, hanem szépen ússzon, azaz amíg eszünkbe nem jut, hogy megvárjuk a cursor-megszakítást. Valójában ettől függetlenül sem teljesen jó a dolog, csak elsőre nem látszik, de mindjárt kiderül, hogy miért is.

A probléma az, hogy a megszakítást az váltja ki, hogy a képernyő raszter (a képernyő frissítés) eléri a számára előre beállított pozíciót, ami alapértelmezetten az utolsó látható képernyősor, azaz a **240**. sor utolsó karaktere. Igen ám, de a videómemória nem csak **240** sor, amennyit alapértelmezetten látunk, hanem igazából **256**, így ha elkezdjük elmozdítani a megjelenítés kezdőcímét, akkor ez a plusz **16** sor is megjelenik a képernyőn. Ez nem is lenne baj (sőt, ezt tudjuk majd kihasználni igazán), de ezzel együtt egyszer csak bekövetkezik, hogy a **240**. sor képernyőn kívülre kerül, ennek a plusz **16** sornak a területére és emiatt nem következik be a megszakítás, mert a raszter meg sosem jár arra, hiszen neki csak a látható képernyőt kell frissítenie. Ez azért is baj, mert a megszakításban azért történnek dolgok - például a billentyűzet kiolvasása -, meg ha egy **HALT** utasítással erre várunk, hogy közvetlenül a képernyőfrissítés megkezdése előtt scrollozzunk, meg rajzoljuk ki, amit akarunk, amitől szép villogásmentes lehet a megjelenítés, akkor látszólag "lefagy" a program, mert a megszakítás nem fog bekövetkezni. Ennél egyébként eggyel még összetettebb a helyzet, de nagyjából ez a problémánk.

A megoldás pedig egyszerű: állítsuk át minden egyes képernyő-eltoláskor a cursor-megszakítás pozícióját is pont ugyanannyival és akkor jóidő, minden rendben lesz, sosem kerül képernyőn kívülre a cursor-megszakítás pozíciója. Hát ez egy szép elmélet, de a gyakorlatban csak nem akart működni. Ezért elkezdtem debuggolni a ***Szánkóverseny*** című játékot, hogy abban mi a búbánatot csinálnak még, amitől mégis működik a függőleges hardver scroll. Meg fogtok lepődni: pont ezt csinálják, azaz minden scroll után átállítják a cursor-megszakítás pozícióját. Szóval jó úton jártam, csak nem volt rajtam sapka :) Vagy legalábbis valamit elbaltáztam.

Mint már említettem, a **CRTC**-vel scrollozás nem egy bonyolult dolog, csak a **12**. és **13**. regiszterébe kell a kezdeti **0** értéket minden scroll-lépésben **64**-el növeli (**64** az egy logikai “karaktersor” mérete, ami **4** képernyősor egyébként). Erre célszerű egy változót vezetni, nevezzük **SCROLL_OFFSET**-nek, ami **0** kezdőértékkel indul és ahogy az előbb említettem, **64**-el növekszik minden scrollozáskor, amíg el nem éri a **4096**-ot, mert akkor nullázzuk. Miért pont **4096**-nál nullázzuk? Mert **256** sor a videomemória, de mivel **4** soronként megy a hardware scroll így egy teljes képernyő körbe-scrollozása az

256 / 4 = 64

lépésben történik meg. Mivel minden lépésben **64**-el növeljük a **SCROLL_OFFSET** változónkat és ahogy kiszámoltuk az előbb, **64** lépés mire körbeér a képernyő, ebből adódik:

64 * 64 = 4096

A **4096**-os értéket már nem küldjük ki, ott már a **0** megy ki, hiszen alapállapotban is **0** a képernyő kezdőcím a **CRTC**-ben, és mivel körbeértünk, így ez ismét az alapállapot, ahonnan ez lehet ismételni a végtelenségig. Egy rövid és gyors módszer a **4096**-os érték nullázására, ha **HL**-ben van a **SCROLL_OFFSET** értéke:

RES 4,H - ami **H** regiszter **4**. bitjét nullázza, ami pont a **4096**-os érték

Hogy legyen értelme is a dolognak, az éppen képernyőn nem levő területre, azaz **241-256**. sorokból az alsó vagy a felső **4** sorba (attól függően, hogy fel vagy le megy-e a scroll), ki kell rajzolni az új grafikákat minden lépésben, hogy már azok scrollozódjanak be. Az adott sor címéhez itt is hozzá kell adni a **SCROLL_OFFSET** változónkat - **16K**-ra levágva a végeredményt, hiszen a videomemória mérete **16K**! -, hogy jó helyre rajzoljunk.

Oké, ez eddig nem túl izgalmas, ez eddig is ment. De mi van a cursor-megszakítás pozíciójával? Az ***Andromeda Software***-es srácok a ***Szánkóverseny*** c. játékban azt találták ki, hogy a **128**. sor utolsó karakterét, mint cursor-megszakítás pozíciót veszik konstansnak és ezt szintén **64**-el növelik minden scrollozás után, hiszen ha a **CRTC** képernyő kezdőcím annyival lett eltolva, akkor ugyanannyival kell a cursor-megszakítás pozíciót is eltolni. Ennek az az előnye, hogy ugyanazt a **SCROLL_OFFSET** változót lehet itt is használni, amivel magát a **CRTC**-s képernyő eltolást vezéreljük, csak hozzá kell adni a fent említett, egyszer papíron kiszámolt és a kódba fixen beírt konstans értéket. Ez a konstans persze számolódhat bármelyik sorból a **0-239** tartományból, ami nekünk éppen megfelel a képernyőre rajzolásunkhoz.

És ennyi, a scroll működik, a megszakítás kiváltódik, az élet szép. :)

Akinek nem lenne meg, így kell kiszámolni a cursor-megszakítás pozíciót, amit a **CRTC 14**. és **15**. regiszterébe kell írni:

CURSOR_POS = (Y DIV 4) * 64 + 63

A képletben **Y** a kívánt sor száma **0-239** között, **DIV** az osztás egész részét jelzi (ha így értelmezhetőbb, akkor **(INT)Y / 4**), a végén szereplő **63** pedig a sor utolsó oszlopa, a **0-63** tartományból.

Érdekesség még, hogy a *Szánkóverseny* kódjában szerepelt a **CRTC 5**-ös regiszterének írására is, ami az *Operációs rendszer* könyv szerint a *"Kiegészítő tv sorok száma"*. Ebben alapértelmezetten **2**-es érték van, de a Szánkóverseny hol **1**-et, hol **0**-át tesz bele. Ennek nem igazán láttam az értelmét, de TV-n még nem teszteltem, pláne nem CRT TV-n, szóval lehet, hogy ott értelmet nyer.

Bertók Zsolt, 2019. december 7.