**TVC-PascalToASM ismertető**

**Működés:**

Ékezetes és különleges karakterek nem megengedettek (csak ASCII 127 alatt jó) – két napom ráment, már utálom az ékezetes karaktereket (PrintChar-al meg lehet jeleníteni az ékezetes karaktereket ASCII kód alapján) !!

Az utolsó END. utáni rész egy az egyben bemásolásra kerül az assembly forrásba! Itt elhelyezhetőek saját ASM-betétek, sprite-ok, pályatérképek stb. A külső assembly fájlban tárolt saját rutinokat is itt lehet INCLUDE-olni pl.: include "fuggv.asm".

**FONTOS:**

1. A kis és nagybetűk nem különböznek!
2. Szorzásnál / összeadásnál figyelni kell a 16bit-es túlcsordulásra mert téves lesz az eredmény!
3. Számításoknál ha használsz konkrét számot akkor a műveleteknél ügyelni kell a sorrendre pl.: Poke(**32768 + i**, i\*2); - rossz eredményt ad!!! **Helyes: Poke(i + 32768, i\*2);**
4. Lehetőleg változókat/konstansokat használjunk!!!
5. Sprite kezelő eljárások csak akkor működnek ha a VIDEO memória a 3. lapon van beállítva (pl.: U0, U1, VID, SYS) !

**Forrásfjál (pas) fordítása CAS-ra:**

Windows:

1. Test1.pas -fájl fordítása:
2. parancssorból → ./TVCPascal.exe Test1.pas (pas→asm)
3. parancssorból → pasmo.exe -v --bin Test1.asm Test1.bin (asm→bin)
4. parancssorból → ./TVCTape.exe Test1.bin Test1.cas (bin→cas)

Linux:

1. wine telepítés (TVCTape fordító miatt kell)
2. pasmo fordító telepítése (pl.: sudo apt install pasmo)
3. terminálból → ./TVCPascal Test1.pas (pas→asm)
4. terminálból → pasmo -v --bin Test1.asm Test1.bin (asm→bin)
5. terminálból → wine TVCTape.exe Test1.bin Test1.cas (bin→cas)

**Fájlok:**

* **error.err** : A sikeresen lefordított pascal forrás lépések. Ha a pascal fordító hibát dob vmiért akkor ebben a fájlban lehet nyomonkövetni hogy meddig jutott a fordításban!
* **libasm.lib** : ASM rutinokat tartalmazó „lib”, szövegszerkesztővel megnyitható és szerkeszthető. Ebből a fájlból húzza be a pascal fordító a parancsokhoz tartozó asm betéteket.

**Program felépítése:**

program <program neve>;

const

var

<constant name 1>=<numeric constant 1>;

<constant name 2>=<numeric constant 2>;

...

<variable name 1.1>,<variable name 1.2>,...: <type 1> <=volume>;

<variable name 2.1>,<variable name 2.2>,...: <type 2> <=volume>;

...

procedure <procedure name 1>;

<statement block>;

procedure <procedure name 2>;

<statement block>;

...

{main program}

begin

<statement block>;

end.

<assembler procedures>

<sprites>

<stb….>

**Megjegyzések:**

A megjegyzéseket kapcsos zárójelek {} közé kell tenni, a programban bárhol elhelyezhetőek, akár több sor is lehet egy pár kapcsos zárójel között!

**Konstansok:**

Csak számok lehetnek 0 – 65535 között.

**Változók és tömbök:**

A változók és tömbök deklarálásakor az fel lehet tölteni értékekkel, például:

var

a: byte = 36;

b: string[10] = 'Nacsasoft';

m: array[2,3] of word = [1,4,6, 3,8,5];

A lenti típusok és tömbök használhatóak:

|  |  |
| --- | --- |
| Byte | 0 és 255 közötti egész szám. |
| Word | 0 és 65535 közötti egész szám. |
| String[n] | n-hosszúságú karakter ahol n = 1 től 255 lehet. |
| Array[1..n1] of Byte  Array[1..n1,1..n2] of Byte | 0–255 egész számok egydimenziós és kétdimenziós tömbjei.  n1, n2 - maximális indexek 1 és 65535 között. |
| Array[1..n1] of Word  Array[1..n1,1..n2] of Word | 0-65535 egész számok egydimenziós és kétdimenziós tömbjei.  n1, n2 - maximális indexek 1 és 65535 között. |
| Array[1..n1] of String[n2] | Karakterláncok egydimenziós tömbjei.  n1 a maximális index 1-től 65535-ig, n2 a string maximális hossza 1-től 255-ig. |

**Számtani kifejezések**

<aritmetikai kifejezés> - aritmetikai kifejezés tetszőleges számú argumentumból és rajtuk végrehajtott műveletből áll. Lehetnek állandók, numerikus változók és numerikus tömbök cellái.

Engedélyezett műveletek:

|  |  |
| --- | --- |
| + | Összeadás |
| - | Kivonás |
| \* | Szorzás |
| / | Egész osztás |
| % | Osztás maradéka pl.: 500 % 138 = 86 mert: 86+138×3=500 |
| () | Műveleti sorrend... |

**Karakterek összefűzése:**

<karakterlánc kifejezés> - egy karakterlánc kifejezés tetszőleges számú argumentumból és rajtuk végrehajtott műveletből állhat. Lehetnek változók, karakter-tömbök cellái, valamint közvetlenül az aposztrófokba zárt karakterek (például 'abc123').

**FONTOS:**

1. **Ha aposztrófokba zárt karaktereket fűz változóval akkor mindig a változóval kell kezdeni!!!! pl.: PrintAt(1,1,nevem + ' -BBB- ' + nev[4] + ' - ' + nev[2]);**
2. **Sztring és szám összefűzése nem megengedett!!**
3. **Numerikus változókon végzett műveletek esetén az eredmény kerül kiírásra! pl.: PrintAt(1,1,bb \* 12); (var bb : byte = 112;) kimenet: 1344**

Engedélyezett műveletek:

|  |  |
| --- | --- |
| + | Összeadás |

**Logikai kifejezések**

<logikai kifejezés> - a logikai kifejezés tetszőleges számú aritmetikai kifejezésből és rajta végrehajtott műveletekből állhat. Karakterlánc-kifejezések használata nem engedélyezett.

**A műveletek prioritásának megváltoztatására szolgáló zárójelek nem támogatottak!!**

Engedélyezett műveletek:

|  |  |  |
| --- | --- | --- |
| = | Egyenlő | 1. prioritás |
| > | Nagyobb | 1. prioritás |
| >= | Nagyobb vagy egyenlő | 1. prioritás |
| < | Kevesebb | 1. prioritás |
| <= | Kevesebb vagy egyenlő | 1. prioritás |
| <> | Nem egyenlő | 1. prioritás |
| And | És | 2. prioritás |
| Or | Vagy | 2. prioritás |
| AndByte(A, B) | Logikai és | A és B byte típusús változók logikai vizsgálatának eredményét adja vissza. |
| OrByte(A, B) | Logikai vagy | A és B byte típusús változók logikai vizsgálatának eredményét adja vissza. |
| XorByte(A, B) | Logikai kizáró vagy | A és B byte típusús változók logikai vizsgálatának eredményét adja vissza. |

**Utasítás blokkok**

<statement block> - egy sorban végrehajtott utasításokat pontosvessző válassza el egymástól, és a BEGIN kulcsszóval kezdődik, valamint az END kulcsszóval zárul. Ha a blokk egyetlen utasításból áll, akkor a BEGIN és az END kulcsszavak elhagyhatók.

**Eljárások**

**Az eljárások csak paraméterek és helyi változók nélkül megengedettek.**

**Pascal parancsok:**

|  |  |
| --- | --- |
| <numeric variable>:=<arithmetic expression>;  <string variable>:=<string expression>; | Érték hozzárendelése egy változóhoz. |
| Case <numeric variable> of  <value 1.1>,<value 1.2>,…: <statement block1>;  <value 2.1>,<value 2.2>,…: <statement block2>;  …  else <statement block N>  end; | Ha a <numerikus változó> <1.1 > vagy <1.2> vagy <1.3> ... (az értékek és a feltételek száma bármilyen  lehet), akkor az <utasításblokk1> kerül végrehajtásra. Ha a <numerikus változó> <2.1> vagy <2.2> vagy <2.3> vagy ..., akkor az <utasításblokk2> kerül végrehajtásra. Ha a feltételek egyike sem teljesül, akkor az ELSE utáni utasításblokk hajtódik végre. Az ELSE megadása opcionális. |
| For <numeric variable>:=<arithmetic expression 1> to <arithmetic expression 2> do <statement block>;  For <numeric variable>:=<arithmetic expression 1> downto <arithmetic expression 2> do <statement block>; | A ciklusmagot egy előre meghatározott számszor hajtjuk végre.  <numerikus változó> - a hurokszámláló aktuális értéke, amely a hurkon belül más operátorok számára elérhető.  <aritmetikai kifejezés 1> - a számláló kezdeti értéke.  <aritmetikai kifejezés 2> - a számláló végső értéke.  Ha a DO -t használjuk, akkor a számláló 1-el növekszik. Ha a DOWNTO -t, akkor 1-el csökken. |
| If <logical expression> then <statement block 1> else <statement block 2>; | Ha a feltétel igaz, akkor az utasításblokk1 végrehajtódik, ha nem igaz, akkor az utasításblokk2 fog végrehajtódni. Az ELSE megadása opcionális. |
| Repeat <statement block> until <logical expression>; | Hátultesztelő ciklus esetén a ciklus magja egyszer mindenképpen végrehajtódik, majd a ciklus végén történik egy feltételvizsgálat, és ha teljesül akkor kilép a ciklusból. |
| While <logical expression> do <statement block>; | A program a ciklusba való belépés előtt megvizsgál egy feltételt - ezt belépési feltételnek nevezzük -, és ha ez teljesül, akkor a ciklusmag végrehajtódik, egyébként nem. A ciklusmag ismételten végrehajtódik, amíg a belépési feltétel teljesül. Ha már nem teljesül, akkor a vezérlés a ciklus utáni utasításra kerül. |

**TVC-Parancsok:**

|  |  |
| --- | --- |
| ArrayClear(<array name>); | Egy tömb törlése, numerikus tömb nulláival vagy üres karakterláncok kitöltésével.  <tömb neve> - a tömb neve paraméterek nélkül. |
| Asm(<procedure name>); | Egy assembly eljárás meghívása. Az assembly eljárást az utolsó END után kell elhelyezni a Pascalban. Amikor kilép az eljárásból a RET paranccsal, a vezérlés átkerül a következő Pascal operátorhoz.  <procedure name> - az eljárás neve nagybetűkkel. |
| Cls | Képernyőtörlés |
| Fill(x,y,n) | Kifestés indítása az adott (x,y) pozícióból az adott (n) színnel. A festés színe nem befolyásolja az eredeti tintaszínt!! |
| Var := Get | Várakozás amíg le nem nyomunk egy gombot majd visszaadja a lenyomott billentyű kódját (byte). |
| Var := GetKeyMatrixState(n) | Billentyűzet mátrix n-edik sorának állapotát adja vissza (byte)  n=0...9 („Operációs rendszer” : függ./13) |
| Graphics(x) | Grafikus felbontás beállítása ahol az x a felbontás:  2 színű üzemmód  4 színű üzemmód  16 színű üzemmód |
| Inkey | A lenyomott billentyű kódját adja vissza, nincs várakozás! |
| InputAtString(x,y,arrInputString) | Szöveg beolvasása az adott (x,y) pozíciótól az arrInputString -tömbbe amit előzőleg deklarálni kell. A tömbbe maximum a string deklarálásakor megadott hossznyi karakter kerül be, ha kevesebb akkor fel lesz töltve szóközökkel, ha több akkor le lesz vágva!  (x = felbontás függő:1- 16, 32, 64, y = 1-24) |
| InputAtNumber(x,y,varWord) | Szám beolvasása az adott (x,y) pozíciótól a varWord – word típusú változóba amit előzőleg deklarálni kell. Ha a beolvasott szám < 0 vagy > 65535 vagy nem egész szám akkor nullát kapunk!  (x = felbontás függő:1- 16, 32, 64, y = 1-24) |
| LineStyle(n) | Vonalstílus beállítása (0-16) Operációs rendszer - 11-es függelék. |
| Pause | Várakozás egy billentyű lenyomására |
| Dat := Peek(n) | N memóriacím tartalmát adja vissza Dat byte típusú változóba. N lehet konstans, változó, szám stb. |
| Plot (x1,y1,x2,y2) | Vonal húzása az aktuális tintaszínnel és stílussal |
| PlotRect(x1,y1,sz,m) | Téglalap rajzolása az aktuális tintaszínnel:  x1,y1 = Bal felső sarok koord.  sz = szélesség ; m = magasság |
| Poke(n1, n2) | N1 memóriacímre beírja az n2-t. |
| PrintPlot(x,y,string)  PrintPlot(x,y,'text')  PrintPlot(x,y,var) | Szöveg, string, tömb kiírása az adott rajzolási pozícióba. (x = 0-1023, y = 0-959)  **Fontos: 'Szöveg' + változó összefűzése nem engedett!! Változó + 'Szöveg' -igen!!**  **Több változó esetén a rajtuk elvégzett műveletek eredménye lesz a kimenet!** |
| PrintAt(x,y,string)  PrintAt(x,y,'text')  PrintAt(x,y,var) | Szöveg vagy string típusú változó kiírása az adott karakter pozícióba (x = felbontás függő:1- 16, 32, 64, y = 1-24)  **Fontos: 'Szöveg' + változó összefűzése nem engedett!! Változó + 'Szöveg' -igen!!**  **Több változó esetén a rajtuk elvégzett műveletek**  **eredménye lesz a kimenet!** |
| PrintChar(x,y,c1) | Definiált karakter kiírása az adott karakter pozícióba.  (x = felbontás függő:1- 16, 32, 64, y = 1-24, c1=ASCII kód 128-223) |
| PrintPlotChar(x , y , c1) | Definiált karakter kiírása az adott rajzolási pozícióba.  (0-1023, y = 0-959, c1=ASCII kód 128-223) |
| PutSprite(spr\_name, x, y, mode) | Sprite kirakása felülírással az x,y -pozícióba a megadott móddal (0=felülíras, 1=XOR).  x,y -byte (x = 0 – 63, y = 0 – 239 ; 0,0-pozíció a képernyő bal felső sarka!)  Sprite adatok a sprite\_name -től kezdődnek a programban az END. után, a cimkének nagybatűnek kell lenni!!  Formátum: spr\_szélesség (byte-ban!), spr\_magasság (pixelsor-ban), spr\_data1, spr\_data2, ….. |
| RandomByte()  RandomWord() | Véletlenszám 1 és 255 között. (byte típus)  Véletlenszám 1 és 65535 között (word típus) |
| SetChar(n , charcode) | Egyedi karakter definiálása. n=karakter azonosító (128-223), charcode byte-típusú tömb (ASCII-kódokkal) a karkter képével (10x8 pontmátrix).  n=változó vagy konstans vagy szám |
| SetInk(n) | Tinta színe :  GR 2, GR 4 : paletta sorszáma  GR 16 : színsorszám |
| SetKeyRepeatRate(n) | Auto-repeat késleltetés 20ms-os egységekben. |
| SetMode(n) | Vonal kereszteződési módja :  0 – Felülírás ; 1 – OR ; 2 – AND ; 3 – XOR |
| SetPaper(n) | Háttér (Papír) színe :  GR 2, GR 4 : paletta sorszáma  GR 16 : színsorszám |
| SetBorder(n) | Keret színe a border kódok alapján! A keret színe független a grafikus üzemmódtól! (lsd.: "Gépi kódú programozás kezdőknek" 5.6 táblázat (204. oldal) |
| SetPalette(n1,n2,n3,n4) | Szín paletta beállítása a 2-es és a 4-es grafikus üzemmódokhoz. Négy és kétszínű üzemmódban beállítja hogy a 16 színből melyik legyen a 0, 1, 2, 3-as, illetve a 0, 1-es sorszámú szín! |
| SoundInit | Hang inicializálása. A programban csak egyszer kell meghívni!! (Igazából egy SoundStop!!) |
| SoundPlaying | Ha = 1 akkor folyik egy hang lejátszása, ha = 0 akkor jöhet a következő hang. |
| SoundPlay(pitch,duration) | Egy hang lejátszása (pitch – word, 0-4095) ; (duration – word, 20msec -ban megadva) |
| SoundStop | Hang kikikapcsolása. |
| SoundVolume(n) | Hangerő beállítása (n=0-15). |
| VideoOn(n) | Videomemória belapozása és az eredeti lapozás lementése.  N = 0 akkor : U0,U1,VID,SYS  N = 1 akkor : U0, U1, VID, U3  N = 2 akkor : U0, U1, U2, U3 |
| VideoOff | Az elmentett (VideoOn()) lapozás visszaállítása. |