Project.....Zeda's Hex Opcode Compilation
Program.....N/A
Author.....Zeda Elnara (ThunderBolt)
E-mail.....xedaelnara@gmail.com
Size.....N/A
Language.....English
Programming.....Assembly :P
Version.....1.30
Last Update.....23 February 2011

Intro

Once upon a time, I was a BASIC programmer. Yep. The end. As a BASIC programmer, I liked to make games and math programs, naturally, but I soon came across an amazing thing... assembly opcodes. Pretty quickly, I realised something horrible. hex opcodes barely exist! Well after two and a half years, I finally learned assembly, but there is a catch. I learned the hex, not the actual assembly like most people. So what does that mean? That means for me to make all of my assembly programs--and you can check some samples on TICalc.org-- I need to make the opcode, so naturally I almost always include it and if I do not include it, I have it on my computer.

So here is what it really boils down to. My goal is to make a compilation of almost all of my opcodes that I have used or have come up with. So I guess here goes. Oh, and if you do not know how to input an opcode, check this video over on TIBD.

Easy Assembly Opcodes

Auto DMS (Revised v1.31) (saves 1 bytes)

3E06FD770AC9

This displays any number in Degrees-Minutes-Seconds automatically.

Auto Fractions (Revised v1.31) (saves 1 bytes)

3E0CFD770AC9

This displays numbers in fraction form if they can be, automatically.

Remove Cursor

FDCB0CE6C9

This stops displaying the blinking cursor.

Run Indicator Off

EF7045C9

This turns the run indicator off

Run Indicator On

EF6D45C9

This turns the run indicator on

Run Indicator Stop

21028A3E01AE77C9

This pauses the runindicator (stops it from moving).

Clear Screen

EF4045C9

Clears the screen currently displayed. This only clears the LCD, not the screen buffer. In other words, if you run this from the homescreen, pressing [2nd][Mode] will restore the screen.

Calc Off

3E01D303C9

Turns off the calc, but it doesn't exit the program. Too bad you have to press on to return.

Screen On

3E03D310C9

Turns the screen on.

Screen Off

3E02D310C9

Turns the screen off.

Alpha Press Off (Revised v1.31) (saves 2 bytes)

3E01FD7712C9

Cancels alpha if it has been pressed or simulated.

Alpha Press On (Revised v1.31) (saves 2 bytes)

3E51FD7712C9

Simulates the alpha key being pressed once.

Alpha Press Lock (Revised v1.31) (saves 2 bytes)

3ED1FD7712C9

Locks the alpha key. Use [2nd][Mode] to escape. In a program, if you want only capital letters used in an input, use this and then command 5 after the command (if you want).

Lowercase Press On (Revised v1.31) (saves 2 bytes)

3E31FD7712C9

Simulates the alpha key being pressed twice. Lowercase does not need to be active to use this.

Lowercase Press On 2 (added v1.31)

3E71FD7712C9

Simulates [2nd][Alpha][Alpha]. Lowercase does not need to be active.

Lowercase Press Lock (Revised v1.31) (saves 2 bytes)

3EB1FD7712C9

Locks lowercase key. Use [2nd][Mode] to escape.
Lowercase does not need to be active to use this.

Indicator Start/Stop

21028A3E01AE77C9

This does not remove the indicator, but it does stop it.

Lowercase Off

FDCB249EC9

This deactivates lowercase typing.

Lowercase On

FDCB24DEC9

This activates lowercase typing. Press Alpha twice.

Lowercase (De)Activate

21148A3E08AE77C9

This will enable or disable lowercase access.

Inverse Text Toggle

21F5893E08AE77C9

This turns inverse text on or off.

Inverse Text On

FDCB05DEC9

Turns inverse text on.

Inverse Text Off

FDCB059EC9

Turns inverse text off.

Screen On/Off

DB10CB6F3E0220013CD310C9

Turns the screen on or off.

Arch/Unarch (Program Rights)

21F8893E02AE77C9

There are some things that a program can't do, like archiving and unarchiving programs, that can be done on the homescreen. Likewise, there are things, like using a For(loop, that can be done in a program, but not on the home screen. This command tricks the calculator into thinking a program is the home screen and the home screen is a program. This command should be used to reset the rights, otherwise, you'll get an error. In other words, if you use it in a program, you need to use it again to tell the calc you are in a program!

QuickKey 1

3A3F84EF8C47EFBF4AC9

This is kind of like getKey, except all of the keys will repeat without a delay.

Key is stored to Ans.

QuickKey 2

3A4584EF8C47EFBF4AC9

Here, the last key press is returned, so even if you aren't pressing a key, a key press is returned. Useful for Pac-Man esque games :P

Key is stored to Ans.

QuickKey 3

3A4584EF8C473A3F84324584EFBF4AC9

This first version detects the current key press and the second version detects the last key press. This version detects a key if it has been pressed between loops. This means that if you tap a key, even if the program isn't using a getkey yet, it will still register.

Key is stored in Ans.

Smart Battery Check

EF6F4C3D280A78FE1E
EF21521808
EFB3503E042001AF
EF8C47EFBF4AC9

This is my BSA (Battery Status All) program. If you are running an 83+, a value of either 0 or 4 is stored in Ans. If you are using an 84+, a value from 0 to 4 is stored to Ans. 0 is the worst and 4 is the best. To be clear, 2 (meaning your batteries are okay) is a possible output if you are using an 84+.

Free Ram

EFE542EF9247EF5641EFBF4AC9

The total free ram (minus the size of this program) is stored to Ans.

Shift screen right

2140930E40060CB7CB1E2310FB0D20F5C9

This will shift the screen 1 pixel right.

Shift screen left

213F960E40060CB7CB162B10FB0D20F5C9

This will shift the screen 1 pixel left.

Shift screen up

214C9311409301F402EDB0EB010C00EF304CC9

This will shift the screen 1 pixel up.

Shift screen down

213396113F9601F402EDB823010C00EF304CC9

This will shift the screen 1 pixel down.

Increase Contrast

2147847EC6D9D834D310C9

This will increase the contrast slightly (unless it is at maximum).

Decrease Contrast

2147847E06D780B8D835D310C9

This will decrease the contrast slightly (unless it is at minimum).

Sleep Mode

3E02D310EF72493E03D310C9

This will turn the screen off, wait for a key press, and then turn the screen back on.

Clear Home Screen

EF4645C9

This will leave the cursor in the same position, but the home screen will be cleared.

Clear Disp

EF5845C9

This will move the cursor back to the start position.

Graph→Screen

214093063EEF7048C9

This neat routine will put the graph screen image on the current screen.

Screen→Graph

214093EF7B4CC9

This will put the current screen image on the graph screen.

ClearHome

2108853E2077110985018000EDB0C9

This will clear the homescreen without actually displaying the home screen. Useful if you are using the graph screen for graphics or whatnot and you need to clear the home screen "behind the scenes."

ClearGraph

214093AF77114193010003EDB0EF6A48C9

This will clear the screen (even of the axes and graph image) without telling the calc to update the screen (like ClrDraw). This also displays the

graph screen.

Disable "Done"

FDCB00AEC9

This removes the "Done" message when the program is finished.

DispASCII

EFD74AEFEF4AEF0445C9

This will display an ASCII letter corresponding to the value in Ans.

DispASCIIList

EFD74A3DC0EB462323C5EF7A41E5EFEF4AEF0445E1C110F1C9

This will output a string of ASCII chars using a list. For example:

```
:65+{7,4,11,11,14
:Asm(prgmDISP
```

That will display "HELLO" on the homescreen. If you change the 65 to 97, it will display "hello" on the homescreen.

Complex Opcodes

These are opcodes that are longer and more involved than the previous opcodes.

CharLength (added v1.30)

```
;This will return the char length of a string instead of the token length.
;For example, where length("sin(ln( would return 2 in BASIC, this code
;returns 7 because there are 7 chars
EFD74AD604C0
6F67EB
4E234623EB
C5D5E51A
EFA342F5
EBEF9445
F1E1D109C1
20020B13
0B1378B120E5
EF9247
EF5641
EFBF4A
C9
```

SCharLength (added v1.30)

```
;This returns the pixel width that a string would use if displayed
;with the small font
EFD74AD604C0
676FEB
4E234623EB
C5D5E51A
EFA342F5
EBEF9445
218D8477EFB44C06004F
F1E1D109C1
20020B13
0B1378B1
20DB
EF9247
EF5641
EFBF4A
C9
```

ListToVars1

```
;This uses a list to store to multiple variables. If the input is:
;     :{0,4,3,12,8,24
;     :Asm(prgmL2R
;Then the result will be:
;   :A is 0
;   :B is 4
;   :C is 3
;   :D is 12
;   :E is 8
;   :F is 24
;*Bonus points to anybody who sees the pattern in that sequence
```

EFD74AFE01C0
1AFE1B38023E1B
EB232347
3E41
C5F5E5F5
EFC541F1
327984
D73003EF0F43
E1010900EDB0
F13CC110E4
C9

ListToVars2

```
;Uses Str1 to name the vars, Ans for the list. If you do:
;      :"ADCZQGB→Str1
;      :{0,1,1,2,3,5,8
;      :Asm(prgmL2V2
;Then result will be:
;    :A is 0
;    :D is 1
;    :C is 1
;    :Z is 2
;    :Q is 3
;    :G is 5
;    :B is 8
```

EFC5413EAA327984
D7300F
3E40061B21EC86
3C772310FB
11EA861313D5
EFD74AFE01C0
1AFE1B38023E1B
EB232347
D11A13D5
C5E5F5
EFC541F1
327984
D73003EF0F43
E1010900EDB0
C110E3
D1C9

ExecCode

```
;Deletes prgmU, then copies Ans to prgmU
;For example:
;      :Input "Code:"Str1
;      :Str1
;      :Asm(prgmEXECCODE
;This will automatically execute prgmU, whether the code is BASIC or assembly
EFD74A
FE04C0
215500            ;55 is the token for "U"
22EC86227984
21F086
EB4E234623
ED43EE86
EDB0
3E05327884
EFF142
3803EFC64F
3E0521EC86
EF3C4C
C9
```

RepeatyKeys

```
;This lets all keys repeat...
;Super fast...
;Not really practical ;D
;Run the code again to deactivate
180A
83473A4584323F8478C9
2100807EFE83
2006AF77323F84C9
11979DEB018000EDB0
DB06210080EF664FC9
```

FastKeys

```
;Keys that normally repeat repeat super fast...
1809
83473E0132428478C9
2100807EFE83
2006AF77323F84C9
11979DEB018000EDB0
DB06210080EF664FC9
```

NumToString

;This will convert the integer part of a real number to a string in Ans. So if you convert -366.17 it will output "-366"

EFD74A
217984
11EC86
7EE60F
470448
2D7E07
3005
0C3EB0121C
2C
2CCDD19D052805CDD19D10F4
C5EF524BD7EFC64F
E1E5EF2743
C11313
21EC86
EDB0
C9
3E30ED6F121CC9

MultiKeys2

;Returns a unique value for any one or two key presses. For example, [mode]+[del] at the same time will return 3191.

017F07210000545C
CB0179D301E5E1DB012FB720083E08856F10ED180E
C506082C0F30025C6510F8C110DD
7CB720016F
6C62
7BB7280A
444D2909290929292919
EF9247
EF5641
EFBF4A
C9

Programs

If you don't have a cable to download programs, here are a few assembly programs that I have made. You can always check TICalc.org for more programs because I generally include the full opcode with the programs (I've only got about 50 asm programs on there :D) As it is, you will need to check TICalc for full documentation of the programs

CopyProg

This will let you copy variables from RAM or archive to other variables using Ans and Str1 for names. You can even copy things like Pic1 to an Appvar or a program to a picture or whatever. It will even let you delete vars.

EFD74AFE04C013131AFE712837
3EAA327984D7D878B7C0
EB4E234623C5
7EE61F77
117884
EDB0
AF12
EFF14278C1D8
EBD306
B728050901090009
4E234623
F5E5C5
3E07D306
EFD74AFE04
EB4E234623
7E
FE71F52003237E0D
E61F77
117884EDB0AF12
EFF1423803EFC64F
F1C8
E1E53A7884
EF704E1313
C1E1F1
EF5480
C9

GetName

This program returns the name of a var as well as the size and archive status. This is useful if you want to create a BASIC memory menu or something and especially useful when used with CopyProg.

EFD74AB72006
210500E51814
1313EBE7E5
EFEF4A
EB29113D9E19
5E2356
E1D5E7
EFEF4A424B
3EFF327A84
E122788403
C5EF444AC1
300A21FA00228E842E011805
0B78B120EA
EF524BD7
EFC94F
118E84210000
1AC64012
23131AB720FA
EF2743
EB4E234623
118E84
DB06F5C5
EBEDB0
EF3E41
EFF142
78C13008
210000F601F51811
EBD306
B7F528050901090009
5E235623EB
EF9247
F12807
2187843E01B677
F1D306
EF5641
EFC24A
C9
05001500
160004FF
07FF08FF
170002FF
03FF0000
0C0001FF

BASIC Sprite

This will let you draw an 8x8 sprite using the speed of assembly. It uses a string in Ans as well an X and Y for coordinates.

EFDA4AEFEF4A
626B
19192929E5
EFE04AEFEF4A
1693CBF3E119E5
EFD74AE1FE04C0
1313
010C08
CDCC9DCDCC9D780600094710F3
EF6A48
C9
1AC6C03002D607ED6F13C9

MultiPics

Lets you manage pictures (including hacked ones) such as storing, recalling (even from archive), archiving and unarchiving. This is directly from the uploaded file. I'm not sure if the mnemonics are correct, but I think they are.

```
 bcall(_RclAns)       ;3   EFD74A
 dec a                ;1   3D
 ret nz               ;1   C0
 inc de               ;1   13
 inc de               ;1   13
 ex de,hl             ;1   EB
 rMov9toOP1           ;1   E7
 push hl              ;1   E5
 bcall(_ConvOP1)      ;3   EFEF4A
 pop hl               ;1   E1
 push af              ;1   F5
 rMov9toOP1           ;1   E7
 bcall(_ConvOP1)      ;3   EFEF4A
 ld h,a               ;1   67
 ld l,96              ;2   2E60
 ld (OP1+1),hl        ;3   227984
 xor a                ;1   AF
 ld (OP1+3),a         ;3   327B84
 pop af               ;1   F1
;=================================================================
;This section is RecallPicX
;=================================================================
 or a                 ;1   B7
 jr nz,26             ;2   201A
  rFindSym             ;1   D7
  ret c                ;1   D8
  ld a,b               ;1   78
  inc de               ;1   13
  inc de               ;1   13
  ex de,hl             ;1   EB
  or a                 ;1   B7
  jr z,4               ;2   2804
   ld de,12             ;3   110C00
   add hl,de            ;1   19
  ld de,plotSScreen    ;3   114093
  ld bc,768            ;3   010003
  bcall(8054h)         ;3   EF5480
  bcall(_GrBufCpy)     ;3   EF6A48
  ret                  ;1   C9
;=================================================================
;This section is StorePicX and DelPicX
;=================================================================
 cp 3                 ;2   FE03
 jr nc,33             ;2   3021
  push af              ;1   F5
```

```
 rFindSym            ;1   D7
 jr c,3              ;2   3803
  bcall(_DelVarArc)   ;3   EFC64F
 pop af              ;1   F1
 cp 2                ;2   FE02
 ret z               ;1   C8
 bcall(_CreatePict)  ;3   EF3343
 ex de,hl            ;1   EB
 ld (hl),0           ;2   3600
 inc hl              ;1   23
 ld (hl),3           ;2   3603
 inc hl              ;1   23
 ex de,hl            ;1   EB
 ld hl,12            ;4   210C00
 bcall(_InsertMem)   ;3   EFF742
 ex de,hl            ;1   EB
 bcall(_SaveDisp)    ;3   EF7B4C
 ret                 ;1   C9
;=================================================================
;This section deals with the archive/unarchive commands
;=================================================================
 sub 3               ;2   D603
 push af             ;1   F5
 rFindSym            ;1   D7
 jr nc,2             ;2   3002
  pop bc              ;1   C1
  ret                 ;1   C9
 ld a,b              ;1   78
 or a                ;1   B7
 jr z,1              ;2   2801
  inc a              ;1   3C
 pop bc              ;1   C1
 xor b               ;1   A8
 ret z               ;1   C8
 bcall(_Arc_Unarc)   ;3   EFD84F
 ret                 ;1   C9
```