Derek Acosta
Cosc - 121, 2017
Lab 2
20 February 2017

I. Abstract

The purpose of this lab was to our further of LC-3 Assembly through an exploration of the PennSim Simulator and argumentation of our previous project. Using what we learned from the previous lab - consisting of reading assembly code, debugging the code, optimizing it, linking files and labels interdependently, and exploring how ("long") jumps and trap routines work - allows us, in this lab, to further our understanding of these concepts by extending them. In this lab, we took the skills aforementioned and understanding of Video RAM, Trap Vectors, KBSR & KBDR, and pointers to further the objectives of this project. The task of this assignment was to implement I/O drivers and traps (x20, x21 and x23) through creating a window display program using memory-mapped, LC-3 graphics display - specifically this assignment dealt with creating a series of service routines within our OS.asm that would facilitate the process of displaying user content to the build in PennSim display. Prior to doing any work on the implementation of these methods, I sat down and created an outline/design plan on how I would structure the program. This required me to visualize my methods initially in C++ and translate them down into assembly. Although a tedious method it served me well moving forward with writing the routines.

The first of these service routines was called _getc which acted as Trap x20 by reading one input character from the keyboard and store it into R0 without echoing the character to the console. Essentially, this task was an exploration of creating our own Trap x20 data function that would allow us to poll for keyboard input when KBSR == 1 and store the content of KBDR into R0. The next service routine involved us implementing a _put_graphic function which acted as Trap x21 permitting the output character in R0 to display to the console or our case VRAM. This routine was very involved and challenged us to keep track and consistently update our pointers within our VRAM for our input character as well as our cursorlocation. This is also the primary location in which we both stored and used the graphic representation of our input characters (A - Z, and a space) at a specific corresponding location and providing them with color. The encoding of these letter involved storing them in memory and out the display in ascii to pixel form. The third service routine involved us implementing a _lil_win acting as Trap x23 by taking the input character stored in R0 and echoing the character back to the console. The routine consisted of creating a window inside our display, which I extended to 90x84 to allow for the correct proportions within our windows, choosing a color for the background and initialize the cursor location to track the placement of our characters. Once the screen was set up, we could then call our _getc and _putc_graphic routines would allow for the content to be printed out to the screen and manipulate when to return a newline, when to stop, and when to clear and refresh the little

window. The finally within all the service routines down, we wrote initializer subroutines that would mold how the OS.asm was to be executed. The ultimate test of our hard work was to feed PennSim our assembly files as well as a text file which the program would read and print out the content to the VRAM. The lil_win subroutine would graphically display the contents of the text file onto the graphics window area.

This project was fundamental in allowing us to explore the nature and quirks within Assembly in the prospect of becoming much more comfortable reading and coding it. Through this lab, I know have a greater appreciation and understanding for the work inherent in creating applications such as word processors.

II.Methods and Observations

A. This part of the project had us create three interconnected subroutines for the purpose of initiating the input character retrieve from the keyboard registers. Additionally, we had to initialize these service routines by creating initializer functions for program to dynamically execute these functions until closed.
→ because resetting PennSim, assembling the several .asm file, loading the .obj files came very tedious to do I wrote a script - attached in the appendix - to do all this
→ PennSim continue feature was very useful in regarding to cutting down the number for times it would take to click the next or step buttons and automated it for you
→ Initially to test I simply loaded the PC where ever my get_c routine was located which help save time
→ I compiled all the graphics by using unix by running this command on the folder:

```
cat *.graphic >> sample.txt
```

And subsequently copying and paste them in my file
→ I tried to discover new types structures to facilitate the programming process such as restructure the way the graphics were written by providing a label to them and storing that into my GDT

Q.1 Create a flow diagram (along with a brief description) for each of the subroutines produced, except for the initializers (probably 3 to 5 depending on your implementation). Provide more detail than the high-level diagram included in the instructions. Feel free to use your favorite application that creates flow diagrams, or feel free to draw this by hand and digitize, or try draw.io, or ... .

1. Response:

_putc_graphic

- store Registers R0 to R7 for future use

↓

- enters preamble which will load the GDT, storing the stack pointer and address of GDT

↓

- use the value character value to determine the index of its corresponding location within the GDT.
- applies for letters and space

↓

LD values back into R1 and R0 and branch to FIRSTTEST

FIRSTTEST

- loads pixel into vram by storing it within charPix pointer

↓

- initializes loop counter and memory by setting 'buff' VRAM in R0

↓

- start window, set registers to store char_rows and set it at the corner (where value is stored in memory.

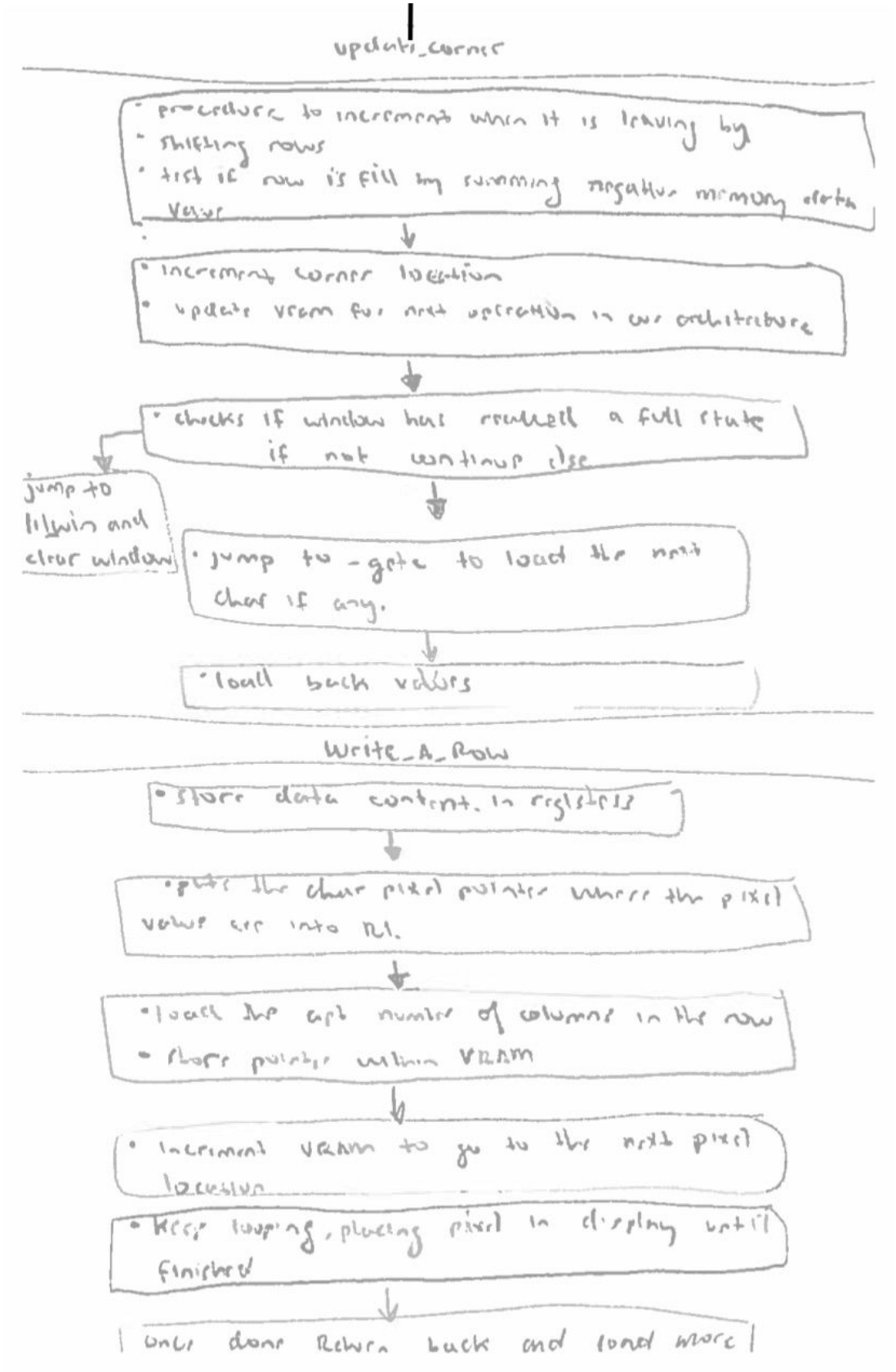
↓

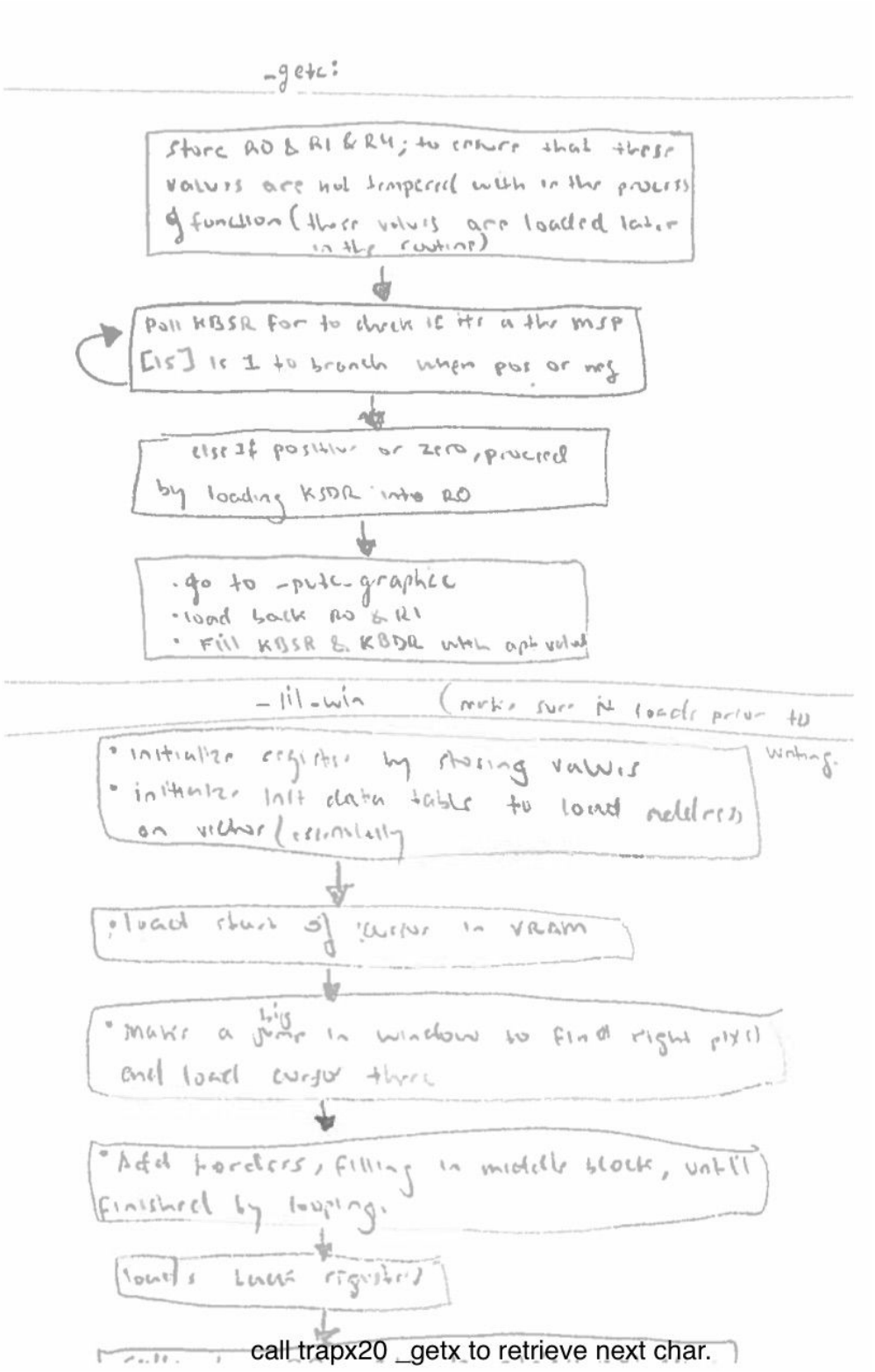
Write loop function

- uses corner stored load enters character, initiating write row function address
- also clearing registers for use

↓

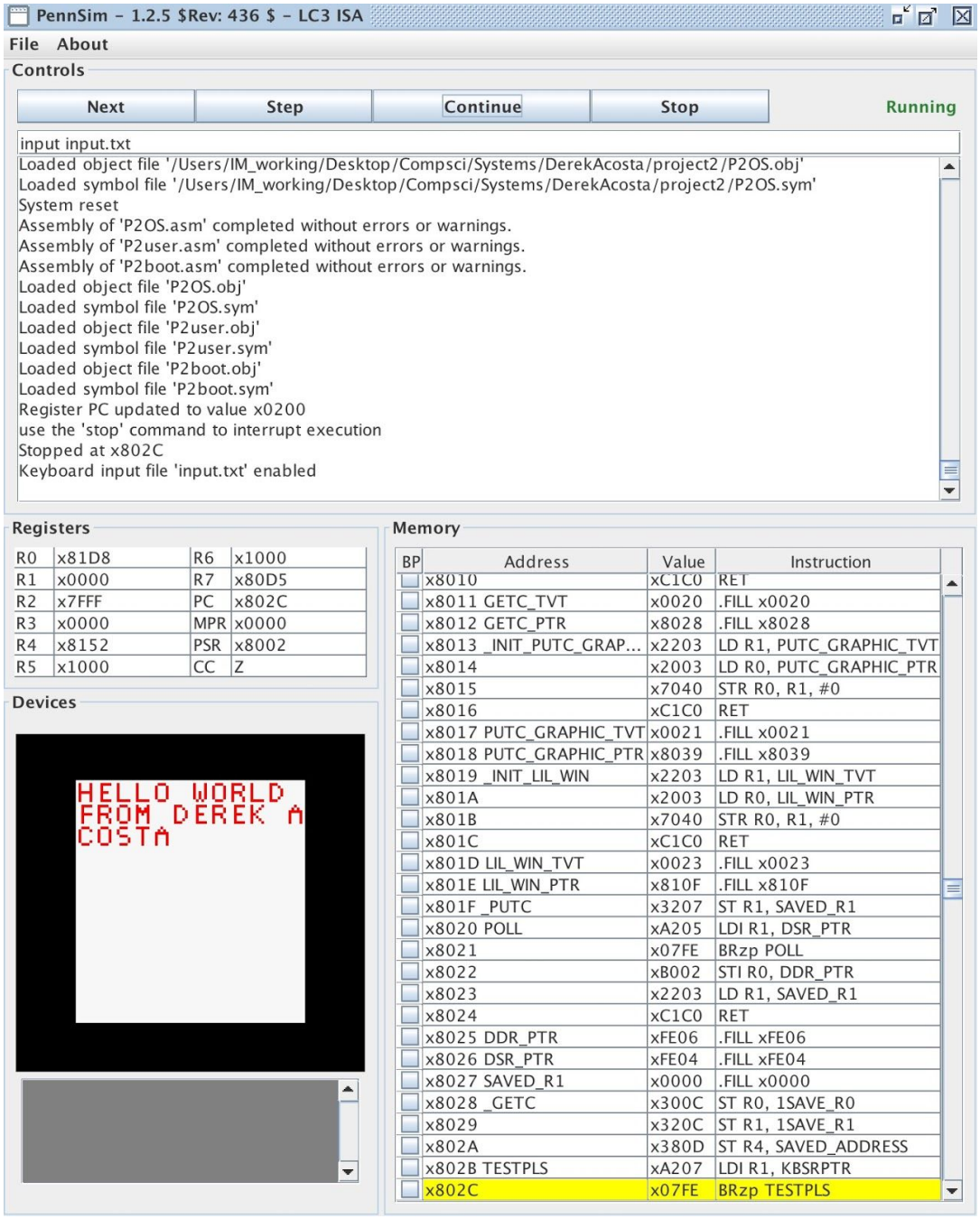
- writes next row in VRAM by summing current by 128


update_corner
• procedure to increment when it is leaving by
• shifting rows
• first if row is fill by running negative memory offset
Value
• increment corner location
• update vram for next operation in our architecture
• checks if window has reached a full state
if not continue else
jump to
fillwin and
clear window
• jump to - gate to load the next
char if any.
• load back values
Write_A_Row
• store data content. in registers
• put the char pixel pointer where the pixel
value are into R1.
• load the apt number of columns in the row
• store pointer within VRAM
• increment VRAM to go to the next pixel
location
• keep looping, placing pixel in display until
finished
once done Return back and load more


-getc:
store R0 & R1 & R4; to ensure that these values are not tempered with in the process of function (these values are loaded later in the routine)
Poll KBSR for to check if it's a the MSP [15] is 1 to branch when pos or neg
else if positive or zero, proceed by loading KSDR into R0
· go to -putc_graphic
· load back R0 & R1
· Fill KBSR & KBDR with apt value
-lil-win (make sure it loads prior to writing.
· initialize registers by storing values
· initialize init data in table to load address on either (essentially
· load start of cursor in VRAM
· make a big jump in window to find right pixel and load cursor there
· Add borders, filling in middle block, until finished by looping.
load s Load registers
call trapx20 _getx to retrieve next char.

Q.2 Provide a screenshot of PennSim with a message successfully displayed in the graphics area.

Response: image below



Q.3 As always, include all code in the Appendix of your report.

Response: check appendix

III. Summary

The report was fully complete and all the questions were addressed and answers. On a few occasion, I had to refactor the code from being very simple to complex but more often than not complex to small because the file would HALT unexpectedly and those errors. These were eventually all taken cared of. Using the simulator was a bit tedious but there were on real issues with the programming itself. Overall the project went well, but required copious amounts of work to fulfill; however, this was a good exposition to computer programming and understanding the complexities with assembly. It reminded me of data structures in regarding to having to write the literal data structures such as a trap in order to communicate amongst I/O devices. It was a fun exploration into seeing how far we, as programmers, have come and where it all began.

IV. Appendix.

See source code below.

Script to assembly and load all files, without having to manually type and locate them

```
reset
as P2OS.asm
as P2user.asm
as P2boot.asm
load P2OS.obj
load P2user.obj
load P2boot.obj
set PC x0200
continue
```

PSboot.asm

```
;;-------------------------------------------------------------
;;-- boot.asm
;;--
;;-- This code runs at power-on because it is loaded to
;;-- x0200 and the PC contains x0200 at power-on.
;;--
;;-- A skeleton booter:
;;-- A typical booter reads OS code into memory from disk,
;;-- then jumps to the OS main(). Our LC3 simulator pretends
```

```
;;-- to do the load from disk by having the simulator load
;;-- the OS code into simulated memory. Thus, all that is
;;-- left for our booter to do is to jump to the OS after
;;-- doing a little initialization for the OS. In the LC3,
;;-- R6 is the Stack Pointer (SP).
;;---------------------------------------------------------------

;;============= .TEXT (boot) ==================================

.ORIG x0200                ;;-- Load to Boot area.

;;------------ boot() -------------
_boot:
   LD  R6 OS_stack        ;;--   Set SP to OS stack area.
   LD  R7 OS_main         ;;--   R7 <=== OS's main() address.
   JMP R7                 ;;--   jump to OS main().
                          ;;--   Never returns.

   OS_stack: .FILL xC000  ;;--   address OS stack area.
   OS_main: .FILL x8000   ;;--   points to OS main().

.END
```

PSuser.asm

```
;;---------------------------------------------------------------
;;-- user.asm
;;--
;;-- A skeleton user-mode program that calls putc() (i.e.,
;;-- TRAP x07). Obviously missing a loop to display the entire
;;-- string. Also, missing is an exit to return to OS (e.g.,
;;-- TRAP x25).
;;---------------------------------------------------------------

;;============= .TEXT (user) ==================================

.ORIG x1000     ;;-- loads to first page of user memory.

   trap x23 ;;-- call _lil_win.

.END
```

PSO2.asm

```
;;-------------------------------------------------------------
;;-- OS.asm
;;--
;;-- A skeleton OS with one service, putc().
;;-- OS starts in main() and begins its intialization phase.
;;-- That phase calls each OS service's initialization routine.
;;-- The init_putc() routine sets putc()'s TRAP vector.
;;-- The OS, having completed its initialization phase, then
;;-- jumps to the user program.
;;--
;;-- putc() displays one character.
;;-- putc() gets its argument character via register R0.
;;-- putc() uses polling to see if the display is ready.
;;-- putc() is assigned the TVT slot x0007 (i.e., TRAP x07).
;;-------------------------------------------------------------


;;============== .TEXT (kernel) ================================
.ORIG x8000                ;;-- Load at start of OS space.

;------- OS main() ----------------
_main:
   JSR _init_lil_win
   JSR _init_putc          ;;--   initialization phase.
   JSR _init_getc
         JSR _init_putc_graphic
   LD  R7 USER_start       ;;--   prepare to jump to user.

   JMP R7                  ;;--   jump to USER space at x1000.

   USER_start: .FILL x1000  ;;--   pointer to USER space.


;;------ init_putc() --------------
_init_putc:

   LD  R1 putc_TVT         ;;--   R1 <=== TVT slot address.
   LD  R0 putc_ptr         ;;--   R0 <=== putc()'s address.
   STR R0 R1 #0            ;;--   write VT: R0 ===> MEM[R1].

   RET                ;;--   return to OS main().
```

```
    putc_TVT:   .FILL x0007  ;;--   points to putc()'s TVT slot.
    putc_ptr:   .FILL _putc  ;;--   points to putc().

;;------ _init_getc(R0) ------------------
_init_getc:
    LD R1 getc_TVT
    LD R0 getc_ptr
    STR R0 R1 #0

    RET

    getc_TVT:   .FILL x0020
    getc_ptr:   .FILL _getc


;;------ _init_putc_graphic(R0) ------------------
_init_putc_graphic:
    LD R1 putc_graphic_TVT
    LD R0 putc_graphic_ptr
    STR R0 R1 #0

    RET

    putc_graphic_TVT:   .FILL x0021
    putc_graphic_ptr:   .FILL _putc_graphic

;;------ _init_lil_win(R0) ------------------
_init_lil_win:

    LD R1 lil_win_TVT
    LD R0 lil_win_ptr
    STR R0 R1 #0

    RET

    lil_win_TVT:        .FILL x0023
    lil_win_ptr:        .FILL _lil_win

;;------ _putc( R0 ) ------------------
_putc:

    ST R1 saved_R1          ;;--   save caller's R1 register.
```

```
   poll:               ;;--  Do
   LDI R1 DSR_ptr      ;;--    read the DSR, R1 <=== DSR;
   BRzp poll           ;;--  until ready, DSR[15] == 1.
   STI R0 DDR_ptr      ;;--  display char, DDR <=== R0.
   LD R1 saved_R1      ;;--  restore caller's R1.

   RET                 ;;--  return to caller.

   DDR_ptr:  .FILL xFE06   ;;--  points to DDR.
   DSR_ptr:  .FILL xFE04   ;;--  points to DSR.
   saved_R1: .FILL x0000   ;;--  space for caller's R1.

;;------ _getc(R0) ------------------
_getc:

        ST      R0 1save_R0
        ST              R1 1save_R1

        ST  R4 saved_address

        TESTPLS:

   poll2:
              LDI     R1 KBSRptr          ; Test for
          BRzp        poll2           ; character input
              LDI     R0 KBDRptr
          BRnzp       _putc_graphic ; Go to the next task

        AFTERTEST:


        BRnzp TESTPLS
        BRnzp MEMORYPROTECT

        LD      R0 1save_R0
        LD      R1 1save_R1

   KBSRptr     .FILL xFE00         ; Address of KBSR
   KBDRptr     .FILL xFE02         ; Address of KBDR
        1save_R0    .BLKW 1
        1save_R1    .BLKW 1

   MEMORYPROTECT:
```

```
        trap x21        ;;Once done JMP to _putc_graphic

            saved_address .FILL GDT                ;;--Constant Global Data table address


;;------ putc_graphic(R0) ------------------



_putc_graphic:

    ;;SAVE EVERYTHING TO BE LOADED FOR THE FUTURE
    ST R0 R0SAVELOOP
    ST R1 R1SAVELOOP
    ST R2 R2SAVELOOP
    ST R3 R3SAVELOOP
    ST R4 R4SAVELOOP
    ST R5 R5SAVELOOP
    ST R6 R6SAVELOOP
    ST R7 R7SAVELOOP


    Entry_:
            BRnzp Preamble_                                ;;-- go to preamble which skips the
instructions
    GDT_add:
            .FILL GDT                ;;--Constant Global Data table address
    Preamble_:
            LD  R4 GDT_add
            LDR R6 R4 #0;; R6 is the stack pointer which has the address of the bottom of the
stack(GDT[0])
            LDR R5 R4 #0;; R5 is the address of the stack bottom which is the base of frame
pointer(GDT[0])

        ;;---Do the subtraction of 56 to get the appropriate values
        ;;--figure out which charachter it is which match the GDT


            ;;---Get the appropriate Value into R0
            ST R1 CORRECTPOSITION
            AND R1 R1 #0
                    LD R1 FIFTYFIVE
                    ADD R1 R0 R1
```

```
        BRp appendLetter
        BRnz appendSpace
           appendSpace:
           AND R1 R1 #0
           ADD R1 R1 #9          ;; for space char
           ADD R1 R1 R4
           AND R0 R0 #0;

        BRnzp getout

        appendLetter:
           ADD R1 R1 R4
           AND R0 R0 #0;
        getout:



           LDR R0 R1 #0
           LD R1 CORRECTPOSITION ;;To get it from the keyboard... This will get the correct
location on the table
BRnzp FIRSTTEST
           CORRECTPOSITION .BLKW 1
           FIFTYFIVE               .FILL #-55
FIRSTTEST:

     LoadPixels:

           TESTG:
           ST R1 THIRTY6
           AND R1 R1 #0
                           LD R1 threesix
                           ADD R1 R1 R4;
           STR R0 R1 #0    ;;--     R0 ===> charPix ( == GDT[11] )
           LD R1 THIRTY6

                   BRnzp TESTVALUES
                           THIRTY6 .BLKW 1
                           threesix .FILL #36
                   TESTVALUES:

           ;;initialize the loop counter and memory
           LDR R0 R4 #6;;--R0=VRAM
           ST R3 SAVER3
```

```
        AND R3 R3 #0
        LDR R3 R4 #-7 ;;gets the value
        ADD R0 R0 R3          ;; Add value;
        ADD R0 R0 #11         ;; to get 22
        ADD R0 R0 #11         ;;
        LD R3 SAVER3          ;;
        BRnzp WINDOWSTART1

WINDOWSTART1:

        STR R0 R4 #8;;--vram=R0


        ;;This gets the number of rows there are

        ;;-PROCEDURE TO FILL VALUE 41
        ST R1 SAVE_R1_1 ;;save whatever is in the value to be used again
        AND R1 R1 #0 ;
    LD R1 STORE41
                ADD R1 R1 R4 ;;ADD R1 to R4 to get GDT[0+41];
        LDR R0 R1 #0;;R0=CHAR_ROWS=9;
        LD R1 SAVE_R1_1    ;; restore it

         BRnzp STORECORNER
                  STORE41     .FILL #41
       SAVE_R1_1   .BLKW 1

STORECORNER:
        NOT R0 R0             ;;2'Ws complement negation
        ADD R0 R0 #1          ;;same thing
        STR R0 R5 #-1         ;;i=-Char_ROWS=-9

    ;;Before the loop, it stores the absolute position
        ST R0 TobeUsed        ;;saves whatever value is in R1
        AND R0 R0 #0          ;; clear
        LDR R0 R4 #6;; get value stored in vram
        STR R0 R4 #-1         ;; Store the corner before it starts
        LD R0 ToBeUsed  ;; load it


;;;;----------------------FUNCTION WRITE LOOP CALL AND RETURN-----------------------------
;;---After the absolute corners gets stored
LOOP0:                                ;;--LOOP to print out any entire charachter
```

```
        LEA R7 RET_0          ;;Saves return value into R7
        ;;--Procedure to FILL VALUE 43
        ST R1 SAVE_R1_2
        AND R1 R1 #0 ;;clears the registers
        LD R1 STORE43
                ADD R1 R1 R4
        LDR R0 R1 #0;;This is the place where the writerow function is located
        LD R1 SAVE_R1_2 ;restore

        BRnzp getNextLevel  ;;unconditional branch to skip the memory allocated here
                STORE43        .FILL #43
                SAVE_R1_2 .BLKW 1


getNextLevel:
        JMP R0                          ;;Put the PC value into R0
        RET_0:

    ;;;----WHERE THE WRITE LOOP FUNCTION ENDS-----------

                                        ;;--write the next row in the vram
        LDR R0 R4 #8;;--R0=vram pointer
        LDR R1 R4 #7;;--R1=128 which goes to next column
        ADD R0 R0 R1          ;;--Sum it up to get to the next level
        STR R0 R4 #8;;--vram=r0

        ;;--another fill value:fill 36
        ST R1 SAVE_R1_3
        AND R1 R1 #0  ;
    LD R1 STORE36_1
                ADD R1 R1 R4
        LDR R0 R1 #0 ;;--function to advance next block for rows (==GDT[36])
        LD R1 SAVE_R1_3

        BRnzp charColumns  ;;unconditional branch to skip the memory allocated here
                STORE36_1     .FILL #36
                SAVE_R1_3   .BLKW 1


charColumns:

        ;;--procedure fill value: 42
```

```
        ST R0 SAVE_R0_0
        AND R0 R0 #0 ;
        LD R0 STORE42
                ADD R0 R0 R4
        LDR R1 R0 #0;;--R1=char_cols (==GDT[42])
        LD R0 SAVE_R0_0

        BRnzp LoadCharPixbuffer
                STORE42         .FILL #42
                SAVE_R0_0   .BLKW 1

    LoadCharPixbuffer:


        ADD R0 R0 R1        ;;--R0=charpix+char_cols

        ;;---PROCEDURE TO FILL VALUE #36
        ST R1 SAVE_R1_5 ;;store the value to be used in R1
        AND R1 R1 #0 ;
                LD R1 STORE36
                ADD R1 R4 R1        ;; R1=GDT[0+36]
        STR R0 R1 #0;;--charpix=R0---Base Register R0 get the address of R4+36 which
                ;;is where is the appropriate memory location for storage
        LD R1 SAVE_R1_5 ;; get the value back

                BRnzp Update_Corner
                SAVE_R1_5  .BLKW 1
                STORE36         .FILL #36

    Update_Corner:

        LDR R3 R5 #-1       ;;Procedure to increment it. its just stored in in an memory location
before x3000
        ADD R3 R3 #1        ;; R3++
        STR R3 R5 #-1       ;; i=R3
        BRn LOOP0           ;;

        ;;--Update the Corner Position whenever it is leaving
        ;;--We're shifting Rows Here
        ST R1 SaveR1Value  ;;Save it for the incrementation Part
        AND R1 R1 #0            ;;AND IT to clear it
        LDR R1 R4 #-3           ;;GET Whatever value is in const row
        ADD R1 R1 #7            ;;Test to see when it is zero
```

```
    BRz CONDITIONZERO        ;;WHEN IT IS ZERO

    ;;-----CONDITION NOT ZERO
    STR R1 R4 #-3                ;;Store it when finished using it
    AND R1 R1 #0                 ;;clear it again
    LD R1 SaveR1Value  ;;Get it back;;
    ST R0 SaveR0Value  ;;--Save the value so that no harm happens
    AND R0 R0 #0                 ;;--Clear R0 to be used
    LDR R0 R4 #-1                ;;--Get the value that is stored here
    ADD R0 R0 #7                 ;;ADD it to advance to the next corner
    STR R0 R4 #-1                ;;Store it into the master memory block
    STR R0 R4 #6          ;;Updates the Vram for the next operation
    LD      R0 SaveR0Value       ;;

    BRnzp RIGHTAFTER

CONDITIONZERO

   ST R0 SaveR0Value

    LDR R1 R4 #-6      ;;GETBACK to init value
    STR R1 R4 #-3                ;;store
    AND R1 R1 #0                 ;;clear
    LDR R1 R4 #-5                ;;prep for large jump


    AND R0 R0 #0                 ;;Clear the register to get the corner location
    LDR R0 R4 #-1                ;;Gets the corner location that is stored check GDT
    ADD R0 R0 R1                 ;;The big JUMP
    STR R0 R4 #-1                ;;Store it into the master memory block
    STR R0 R4 #6          ;;Updates the Vram for the next operation


    ST R2 SaveR2Value  ;;Save the value
    AND R2 R2 #0                 ;;Clears it
    LDR R2 R4 #-2                ;;Loads to check to see if it reached the end
    ADD R2 R2 #9                 ;;CHECK
    BRz THEEND         ;;WINDOW IS FULL OR SHOULD BE
    STR R2 R4 #-2                ;;DONE


    BRnzp GO_ON                       ;;NOT THE END continue
```

```
   THEEND:
      LDR R0 R4 #-8      ;;retores cursors and all other values
      STR R0 R4 #6       ;;once the table is full
      LDR R0 R4 #-9
      STR R0 R4 #-2

      LD R1 userADD
      JMP R1

      GO_ON:                              ;;KEEP ON GOING



RIGHTAFTER
      LD       R0 SaveR0Value       ;;Saves it
      LD  R1 SaveR1Value ;;TEST IT

   trap x20          ;;JUMP TO getc FOR get other chars



      BRnzp UpDateCorner
      SaveR0Value .BLKW 1
      SaveR1Value .BLKW 1
      SaveR2Value .BLKW 1 ;;TEST TO SEE IF IT FINISHES AFTER BIG JUMP

upDateCorner:

      LD R0 R0SAVELOOP
      LD R1 R1SAVELOOP
      LD R2 R2SAVELOOP
      LD R3 R3SAVELOOP
      LD R4 R4SAVELOOP
      LD R5 R5SAVELOOP
      LD R6 R6SAVELOOP
      LD R7 R7SAVELOOP

      BRnzp AFTERTEST

      R0SAVELOOP .BLKW 1
      R1SAVELOOP .BLKW 1
      R2SAVELOOP .BLKW 1
      R3SAVELOOP .BLKW 1
```

```
        R4SAVELOOP .BLKW 1
        R5SAVELOOP .BLKW 1
        R6SAVELOOP .BLKW 1
        R7SAVELOOP .BLKW 1

                ;;----preserving----------
   useradd          .fill x1000
   ToBeUsed                 .BLKW 1
   SAVER3                           .BLKW 1

;;------SUBROUTINES--------

Write_A_ROW:

   ;;safely storing the registers
        ST R0 R0SAVELOOP
        ST R1 R1SAVELOOP
        ST R3 R3SAVELOOP

        LDR R0 R4 #8;;--put the vram pointer to dereference R0


   ;;----PROCEDURE TO FILL 36 INTO THE LOOP

   ;;retrieve charPix
        ST R0 SAVE_R0_L
        AND R0 R0 #0  ;
        LD R0 STORE36_3
        ADD R0 R0 R4
        LDR R1 R0 #0;;--put the char pixel pointer-where the pixel values are into register R1
        LD R0 SAVE_R0_L

        BRnzp InsideLoop1
        STORE36_3 .FILL #36
        SAVE_R0_L .BLKW 1
InsideLoop1:

   ;;----PROCEDURE TO FILL 42 INTO THE LOOP

        ST R0 SAVE_R0_LA
        AND R0 R0 #0;
                        LD R0 STORE42_F
        ADD R0 R0 R4 ;;get the appropriate values
```

```
        LDR R3 R0 #0;;--This get the number of columns in a row
        LD R0 SAVE_R0_LA

                BRnzp anotherBranch
                STORE42_F .FILL #42
                SAVE_R0_LA .BLKW 1

        anotherBranch
        NOT R3 R3           ;;get 2's complement
        ADD R3 R3 #1        ;;
        LOOP1:                      ;;
        LDR R2 R1 #0;;R2=*charachter pixel pointer values
        STR R2 R0 #0;;R2=*vram
        ADD R1 R1 #1        ;;incrementation of charachter pixel pointer to get to the next one
        ADD R0 R0 #1        ;;incrementation of vram to go to the next pixel location
        ADD R3 R3 #1        ;;R3++
          BRn LOOP1                 ;;until its finished

      LD R0 R0SAVELOOP
        LD R1 R1SAVELOOP
        LD R3 R3SAVELOOP

      RET                           ;;return back JMP 7

        ;SAVE_R0_L .BLKW 1
      ;SAVE_R0_LA .BLKW 1

   ;;-------creating the actual windows for text bounds
_lil_win:

   Entry BRnzp WindowStart

   INITADDRESS .FILL INIT_DATA_TABLE ;;Get the address of the table

   ;;R1 initialized to cursor

   WindowStart:

   ;;Use callee save to save Values for the registers we are using
        ST R1 WinSaveR1
      ST R4 WinSaveR4
      ST R5 WinSaveR5
      ST R2 WinSaveR2 ;;SAVE R2 to LOAD THE PIXEL
```

```
        AND R1 R1 #0        ;;clear R1
        AND R4 R4 #0        ;;clear R4
        AND R5 R5 #0        ;;clear R5
        LD R4 INITADDRESS ;;R4 now has the starting address of the data table
        LDR R1 R4 #5    ;;Start of the Cursor in the VRAM
        AND R2 R2 #0        ;;clear
        LDR R2 R4 #4;;Load the color



    ;;---- This part adds the value to the cursor

        ST R3 WinSaveR3     ;;Save the value of R3 to get the big addition
        AND R3 R3 #0        ;;Clears it to be used again
        LDR R3 R4 #1;;Get the value for the big addition
        ADD R1 R1 R3        ;;Increment the cursor straight to the starting address of the
window
        LD R3 WinSaveR3     ;;Get it back now that R3 is not used again
        ;;-----Initialize the counter to -90
        LDR R5 R4 #3;;R5=-90



    MiddleBLOCK:

        ;;-----Do the decrementation after

                ;;add 22
        ADD R1 R1 #11       ;;Add the cursor so that it increments by the small blocks
        ADD R1 R1 #11       ;;Add the cursor so that it increments by the small blocks
        ST R3 WinSaveR3 ;;R3 is used as a counter to traverse through the columns
        AND R3 R3 #0        ;;Clears the columns
        LDR R3 R4 #2;;Get the counter value of -84

    InnerBLOCK:

        STR R2 R1 #0;;Create Pixel Incorrect Storage mechanism
        ADD R1 R1 #1        ;;Increment cursor
        ADD R3 R3 #1        ;;row++
        BRn InnerBlock      ;; branch until reach 0
        AND R3 R3 #0                ;;Clears it
        LDR R3 R4 #2        ;;Resets it to -84
```

```
        ADD R1 R1 #11       ;;Adds it so that reaches the far edge
        ADD R1 R1 #11       ;;Adds it so that reaches the far edge
        ADD R5 R5 #1        ;;Increment so that it does it 90 times
        BRn MiddleBLOCK ;; LOOP UNTIL FINISH




    ;;---Restore the register for additional usage
    ;;-----preserving values
        LD R3 WinSaveR3
        LD R5 WinSaveR5
        LD R4 WinSaveR4
        LD R2 WinSaveR2


    ;;---Final Save for the final big addition
        ST R3 WinSaveR3     ;;Save the value of R3 to get the big addition
        AND R3 R3 #0        ;;Clears it to be used again
        LDR R3 R4 #1;;Get the value for the bottom allocation
        ADD R1 R1 R3        ;;shift cursor to starting pixel of window
        LD R3 WinSaveR3     ;;Get it back now that R3 is not used again
        LD R1 WinSaveR1 ;;Final SAVE should get it

       trap x20       ;;ONCE this is done jump back to getc and being polling


    BRnzp PROTECTION
    WinSaveR1 .BLKW 1
    WinSaveR2 .BLKW 1 ;; We have to load the pixel to Store it... Way I have is WAY TOO
DIRECT
    WinSaveR4 .BLKW 1
    WinSaveR3 .BLKW 1
    WinSaveR5 .BLKW 1
    PROTECTION
    HALT ;;STOPS AS TEST

    ;;;;;;;;;-----Initialization Data Table----------------------------------------------
    INIT_DATA_TABLE:
    .FILL x2000     ;;--- ( 0)  InitializationBottom
    .FILL #2176         ;;--- ( 1)        Number of rows for allocating block in memory
    .FILL #-84          ;;--- ( 2)        COLUMNS: The window that's actually initialized
```

```
   .FILL #-89              ;;----( 3)  ROWS: The window that's actually initialized - set to precise fit
   .FILL xffff             ;;----( 4)          WHITE
   .FILL xC000     ;;--- ( 5)  const VRAM for direct way to load into initialization Table



;;-------------END OF _lil_win


         ;;        tables allows to a range of -33 to 32 however we take a offset
   ;;------ _GDT() ------------------
   ;;---offset      name             description
   .FILL #-99              ;;--- (-9)                    HOWDEEPCOLUMN
   .FILL xC000     ;;----(-8)
         .FILL #2176             ;;----(-7)
         .FILL #-84              ;;----(-6)                GETBACK VALUE
         .FILL #947              ;;----(-5)                JUMP Columns
         .BLKW 1                         ;;----(-4)                Variable Name
         .FILL #-84              ;;----(-3)                Const row counter
         .FILL #-99              ;;--- (-2)                HOWDEEPCOLUMN
         .BLKW 1                         ;;--- (-1)                CornerColumnPosition
GDT:
   .FILL x1000     ;;--- (  0)  const STACKBOT     Bottom of Stack
   .FILL x7FFF     ;;--- (  1)  const WHITE
   .FILL x0000     ;;--- (  2)  const BLACK
   .FILL x7C00     ;;--- (  3)  const RED
   .FILL x03E0     ;;--- (  4)  const GREEN
   .FILL x001F     ;;--- (  5)  const BLUE
   .FILL xC000     ;;--- (  6)  const VRAM         start of VRAM
   .FILL x0080     ;;--- (  7)  const COLS         columns per VRAM row
   .BLKW 1         ;;--- (  8)  var   vram         pointer into VRAM

         ;;------- THE LOCATIONS OF ALL THE Charachter Pixel Blocks----------------------
   .FILL charSPACE   ;;--- ( 9)  const charSPACE       address of 'SPACE' font buffer
   .FILL charA   ;;--- ( 10)  const charA       address of 'A' font buffer
   .FILL charB   ;;--- ( 11)  const charB       address of 'B' font buffer
   .FILL charC   ;;--- ( 12)  const charC       address of 'C' font buffer
   .FILL charD   ;;--- ( 13)  const charD       address of 'D' font buffer
   .FILL charE   ;;--- ( 14)  const charE       address of 'E' font buffer
   .FILL charF   ;;--- ( 15)  const charF       address of 'F' font buffer
   .FILL charG   ;;--- ( 16)  const charG       address of 'G' font buffer
   .FILL charH   ;;--- ( 17)  const charH       address of 'H' font buffer
```

```
.FILL charI   ;;--- ( 18)  const charI       address of 'I' font buffer
.FILL charJ   ;;--- ( 19)  const charJ       address of 'J' font buffer
.FILL charK   ;;--- ( 20)  const charK       address of 'K' font buffer
.FILL charL   ;;--- ( 21)  const charL       address of 'L' font buffer
.FILL charM   ;;--- ( 22)  const charM       address of 'M' font buffer
.FILL charN   ;;--- ( 23)  const charN       address of 'N' font buffer
.FILL charO   ;;--- ( 24)  const charO       address of 'P' font buffer
.FILL charP   ;;--- ( 25)  const charP       address of 'P' font buffer
.FILL charQ   ;;--- ( 26)  const charQ       address of 'Q' font buffer
.FILL charR   ;;--- ( 27)  const charR       address of 'R' font buffer
.FILL charS   ;;--- ( 28)  const charS       address of 'S' font buffer
.FILL charT   ;;--- ( 29)  const charT       address of 'T' font buffer
.FILL charU   ;;--- ( 30)  const charU       address of 'U' font buffer
.FILL charV   ;;--- ( 31)  const charV       address of 'V' font buffer
.FILL charW   ;;--- ( 32)  const charW       address of 'W' font buffer
.FILL charX   ;;--- ( 33)  const charX       address of 'X' font buffer
.FILL charY   ;;--- ( 34)  const charY       address of 'Y' font buffer
.FILL charZ   ;;--- ( 35)  const charZ       address of 'Z' font buffer

      ;;--------Other Important Variables---------------------------------
.BLKW 1         ;;--- ( 36)  var charPix        pointer into char pixel buffer
.BLKW 1         ;;--- ( 37)  var char           pointer to char buffer
.BLKW 1         ;;--- ( 38)  var vram_row       cursor row in VRAM
.BLKW 1         ;;--- ( 39)  var vram_col       cursor column in VRAM
.FILL x0000     ;;--- ( 40)  const 0           value 0
.FILL x0009     ;;--- ( 41)  const CHAR_ROWS    num rows per char buffer
.FILL x0007     ;;--- ( 42)  const CHAR_COLS    num cols per char buffer
.FILL Write_A_ROW ;;--- ( 43)  const writeRow     function pointer
                ;;--------The Table of ASCII values---------------------------------
.FILL x0041     ;;--- ( 44)   const Ascii_A     ASCII code for 'A'
.FILL x0042     ;;--- ( 45)     .....           ASCII code for 'B'
      .FILL x0043     ;;--- ( 46)        .....          ASCII code for 'C'
.FILL x0044     ;;--- ( 47)     .....           ASCII code for 'D'
      .FILL x0045     ;;--- ( 48)        .....          ASCII code for 'E'
.FILL x0046     ;;--- ( 49)     .....           ASCII code for 'F'
      .FILL x0047     ;;--- ( 50)        .....          ASCII code for 'G'
.FILL x0048     ;;--- ( 51)     .....           ASCII code for 'H'
      .FILL x0049     ;;--- ( 52)        .....          ASCII code for 'I'
.FILL x004A     ;;--- ( 53)     .....           ASCII code for 'J'
      .FILL x004B     ;;--- ( 54)        .....          ASCII code for 'K'
.FILL x004C     ;;--- ( 55)     .....           ASCII code for 'L'
      .FILL x004D     ;;--- ( 56)        .....          ASCII code for 'M'
.FILL x004E     ;;--- ( 57)     .....           ASCII code for 'N'
```

```
        .FILL x004F     ;;--- ( 58)      ......          ASCII code for 'O'
    .FILL x0050     ;;--- ( 59)      ......          ASCII code for 'P'
        .FILL x0051     ;;--- ( 60)      ......          ASCII code for 'Q'
        .FILL x0052     ;;--- ( 61)      ......          ASCII code for 'R'
    .FILL x0053     ;;--- ( 62)      ......          ASCII code for 'S'
        .FILL x0054     ;;--- ( 63)      ......          ASCII code for 'T'
    .FILL x0055     ;;--- ( 64)      ......          ASCII code for 'U'
        .FILL x0056     ;;--- ( 65)      ......          ASCII code for 'V'
    .FILL x0057     ;;--- ( 66)      ......          ASCII code for 'W'
        .FILL x0058     ;;--- ( 67)      ......          ASCII code for 'X'
    .FILL x0059     ;;--- ( 68)      ......          ASCII code for 'Y
        .FILL x0060     ;;--- ( 69)      ......          ASCII code for 'Z'
    .FILL x0032     ;;--- ( 70)      ......        ASCII code for 'SPACE'

charSPACE:
    SPACEROW0: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    SPACEROW1: .FILL x7fff
            .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    SPACEROW2: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    SPACEROW3: .FILL x7fff
            .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
```

```
    SPACEROW4: .FILL x7fff
       .FILL x7fff
       .FILL x7fff
       .FILL x7fff
       .FILL x7fff
       .FILL x7fff
       .FILL x7fff
    SPACEROW5: .FILL x7fff
          .FILL x7fff
       .FILL x7fff
       .FILL x7fff
       .FILL x7fff
       .FILL x7fff
       .FILL x7fff
    SPACEROW6: .FILL x7fff
       .FILL x7fff
       .FILL x7fff
       .FILL x7fff
       .FILL x7fff
       .FILL x7fff
       .FILL x7fff
    SPACEROW7: .FILL x7fff
       .FILL x7fff
       .FILL x7fff
       .FILL x7fff
       .FILL x7fff
       .FILL x7fff
       .FILL x7fff
    SPACEROW8: .FILL x7fff
       .FILL x7fff
       .FILL x7fff
       .FILL x7fff
       .FILL x7fff
       .FILL x7fff
       .FILL x7fff
charA:
    AROW0: .FILL x7fff
       .FILL x7fff
       .FILL x7fff
       .FILL x7fff
       .FILL x7fff
       .FILL x7fff
       .FILL x7fff
```

```
AROW1: .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
AROW2: .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
AROW3: .FILL x7fff
    .FILL x7c00
    .FILL x7c00
    .FILL x7c00
    .FILL x7c00
    .FILL x7c00
    .FILL x7fff
AROW4: .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
AROW5: .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
AROW6: .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
AROW7: .FILL x7fff
```

```
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    AROW8: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
charB:
    BROW0: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    BROW1: .FILL x7fff
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
    BROW2: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    BROW3: .FILL x7fff
        .FILL x7c00
        .FILL x7cff
        .FILL x7cff
        .FILL x7cff
        .FILL x7c00
        .FILL x7fff
    BROW4: .FILL x7fff
```

```
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
    BROW5: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    BROW6: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    BROW7: .FILL x7fff
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
    BROW8: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
charC:
    CROW0: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    CROW1: .FILL x7fff
```

```
    .FILL x7fff
    .FILL x7c00
    .FILL x7c00
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
CROW2: .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
CROW3: .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
CROW4: .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
CROW5: .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
CROW6: .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
CROW7: .FILL x7fff
    .FILL x7fff
```

```
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
    CROW8: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
charD:
    DROW0: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    DROW1: .FILL x7fff
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    DROW2: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
    DROW3: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    DROW4: .FILL x7fff
        .FILL x7c00
```

```
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    DROW5: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    DROW6: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
    DROW7: .FILL x7fff
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    DROW8: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
charE:
    EROW0: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    EROW1: .FILL x7fff
        .FILL x7fff
```

```
    .FILL x7c00
    .FILL x7c00
    .FILL x7c00
    .FILL x7c00
    .FILL x7fff
EROW2: .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
EROW3: .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
EROW4: .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7c00
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
EROW5: .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
EROW6: .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
EROW7: .FILL x7fff
    .FILL x7fff
    .FILL x7c00
```

```
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7fff
    EROW8: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
charF:
    FROW0: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    FROW1: .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7fff
    FROW2: .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    FROW3: .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    FROW4: .FILL x7fff
        .FILL x7fff
        .FILL x7c00
```

```
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7fff
    FROW5: .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    FROW6: .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    FROW7: .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    FROW8: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
charG:
    GROW0: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    GROW1: .FILL x7fff
        .FILL x7fff
        .FILL x7c00
```

.FILL x7c00
.FILL x7c00
.FILL x7fff
.FILL x7fff
GROW2: .FILL x7fff
.FILL x7c00
.FILL x7fff
.FILL x7fff
.FILL x7fff
.FILL x7c00
.FILL x7fff
GROW3: .FILL x7fff
.FILL x7c00
.FILL x7fff
.FILL x7fff
.FILL x7fff
.FILL x7fff
.FILL x7fff
GROW4: .FILL x7fff
.FILL x7c00
.FILL x7fff
.FILL x7c00
.FILL x7c00
.FILL x7c00
.FILL x7fff
GROW5: .FILL x7fff
.FILL x7c00
.FILL x7fff
.FILL x7fff
.FILL x7fff
.FILL x7c00
.FILL x7fff
GROW6: .FILL x7fff
.FILL x7c00
.FILL x7fff
.FILL x7fff
.FILL x7fff
.FILL x7c00
.FILL x7fff
GROW7: .FILL x7fff
.FILL x7fff
.FILL x7c00
.FILL x7c00

```
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
    GROW8: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
charH:
    HROW0: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    HROW1: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    HROW2: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    HROW3: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    HROW4: .FILL x7fff
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
```

```
        .FILL x7c00
        .FILL x7c00
        .FILL x7fff
    HROW5: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    HROW6: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    HROW7: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    HROW8: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
charI:
    IROW0: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    IROW1: .FILL x7fff
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
```

```
    .FILL x7c00
    .FILL x7c00
    .FILL x7fff
IROW2: .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
IROW3: .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
IROW4: .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
IROW5: .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
IROW6: .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
IROW7: .FILL x7fff
    .FILL x7c00
    .FILL x7c00
    .FILL x7c00
    .FILL x7c00
```

```
        .FILL x7c00
        .FILL x7fff
    IROW8: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
charJ:
    JROW0: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    JROW1: .FILL x7fff
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7fff
    JROW2: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    JROW3: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    JROW4: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
```

```
        .FILL x7fff
        .FILL x7fff
    JROW5: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    JROW6: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    JROW7: .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    JROW8: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
charK:
    KROW0: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    KROW1: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
```

```
    .FILL x7c00
    .FILL x7fff
KROW2: .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
KROW3: .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
KROW4: .FILL x7fff
    .FILL x7c00
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
KROW5: .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
KROW6: .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
KROW7: .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
```

```
        .FILL x7fff
    KROW8: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
charL:
    LROW0: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    LROW1: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    LROW2: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    LROW3: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    LROW4: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
```

```
        .FILL x7fff
    LROW5: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    LROW6: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    LROW7: .FILL x7fff
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7fff
    LROW8: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
charM:
    MROW0: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    MROW1: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
```

```
    .FILL x7fff
MROW2: .FILL x7fff
    .FILL x7c00
    .FILL x7c00
    .FILL x7fff
    .FILL x7c00
    .FILL x7c00
    .FILL x7fff
MROW3: .FILL x7fff
    .FILL x7c00
    .FILL x7c00
    .FILL x7c00
    .FILL x7c00
    .FILL x7c00
    .FILL x7fff
MROW4: .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
MROW5: .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
MROW6: .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
MROW7: .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
```

```
    MROW8: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
charN:
    NROW0: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    NROW1: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    NROW2: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    NROW3: .FILL x7fff
        .FILL x7c00
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    NROW4: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
```

```
    NROW5: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7c00
        .FILL x7fff
    NROW6: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    NROW7: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    NROW8: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
charO:
    OROW0: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    OROW1: .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
```

```
OROW2: .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
OROW3: .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
OROW4: .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
OROW5: .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
OROW6: .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
OROW7: .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7c00
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
OROW8: .FILL x7fff
```

```
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
charP:
    PROW0: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    PROW1: .FILL x7fff
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
    PROW2: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    PROW3: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    PROW4: .FILL x7fff
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
    PROW5: .FILL x7fff
```

```
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    PROW6: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    PROW7: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    PROW8: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
charQ:
    QROW0: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    QROW1: .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
    QROW2: .FILL x7fff
```

```
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
QROW3: .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
QROW4: .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
QROW5: .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
QROW6: .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7c00
    .FILL x7fff
QROW7: .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7c00
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
QROW8: .FILL x7fff
    .FILL x7fff
```

```
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
charR:
    RROW0: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    RROW1: .FILL x7fff
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
    RROW2: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    RROW3: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    RROW4: .FILL x7fff
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
    RROW5: .FILL x7fff
        .FILL x7c00
```

```
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
    RROW6: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    RROW7: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    RROW8: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
charS:
    SROW0: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    SROW1: .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
    SROW2: .FILL x7fff
        .FILL x7c00
```

```
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
SROW3: .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
SROW4: .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7c00
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
SROW5: .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
SROW6: .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
SROW7: .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7c00
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
SROW8: .FILL x7fff
    .FILL x7fff
    .FILL x7fff
```

```
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
charT:
    TROW0: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    TROW1: .FILL x7fff
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7fff
    TROW2: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    TROW3: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    TROW4: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    TROW5: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
```

```
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    TROW6: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    TROW7: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    TROW8: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
charU:
    UROW0: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    UROW1: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    UROW2: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
```

```
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
UROW3: .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
UROW4: .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
UROW5: .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
UROW6: .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
UROW7: .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7c00
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
UROW8: .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
```

```
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
charV:
    VROW0: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    VROW1: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    VROW2: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    VROW3: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    VROW4: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    VROW5: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
```

```
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    VROW6: .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
    VROW7: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    VROW8: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
charW:
    WROW0: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    WROW1: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    WROW2: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
```

```
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
WROW3: .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
WROW4: .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
WROW5: .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
WROW6: .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
WROW7: .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
WROW8: .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
```

```
        .FILL x7fff
        .FILL x7fff
charX:
    XROW0: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    XROW1: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    XROW2: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    XROW3: .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
    XROW4: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    XROW5: .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7c00
```

```
        .FILL x7fff
        .FILL x7fff
    XROW6: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    XROW7: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    XROW8: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
charY:
    YROW0: .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    YROW1: .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    YROW2: .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7c00
```

```
    .FILL x7fff
    .FILL x7fff
YROW3: .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
YROW4: .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
YROW5: .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
YROW6: .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
YROW7: .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
YROW8: .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
```

```
        .FILL x7fff
charZ:
    ZROW0:
                    .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    ZROW1:
                    .FILL x7fff
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7c00
        .FILL x7fff
    ZROW2:
                    .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
    ZROW3:
                     .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
    ZROW4:
                    .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7c00
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
    ZROW5:
```

```
                .FILL x7fff
    .FILL x7fff
    .FILL x7c00
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
    .FILL x7fff
ZROW6:
                .FILL x7fff
                .FILL x7c00
                .FILL x7fff
                .FILL x7fff
                .FILL x7fff
    .FILL x7fff
    .FILL x7fff
ZROW7:
  .FILL x7fff
  .FILL x7c00
  .FILL x7c00
  .FILL x7c00
  .FILL x7c00
  .FILL x7c00
  .FILL x7fff
ZROW8:
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
        .FILL x7fff
```