Construcción de MTX 4.0

# Construcción del cargador de arranque (bootloader)

make booter

# Construcción del núcleo o kernel del sistema (mtx)

make mtx

# Construcción de una imagen de disco de arranque (disco floopy) mFDimage

make mFDimage

# Ejecución del sistema MTX 4.0 sobre qemu-system-i386 (emulador de una PC)

make run0

o bien teclear el comando:

qemu-system-i386 –fda mFDimage –no-fd-bootchk

El archivo makefile utilzado es el siguiente:

booter:mtxFS/bs.s mtxFS/bc.c

@echo "compiling booter ..."

as86 -o bs.o mtxFS/bs.s

bcc -o bc.o -c -ansi mtxFS/bc.c

@echo "linking booter ....."

ld86 -d -o booter bs.o bc.o /usr/lib/bcc/libc.a

mtx:ts.s t.c

as86 -o ts.o ts.s

bcc -o t.o -c -ansi t.c

ld86 -d -o mtx ts.o t.o mtxlib /usr/lib/bcc/libc.a

mFDimage:booter mtx

cp -v mtx ./mount\_point/boot/

genext2fs -b 1440 -d ./mount\_point/ mFDimage

dd if=booter of=mFDimage bs=512 count=2 conv=notrunc

run0:mFDimage

qemu-system-i386 -fda mFDimage -no-fd-bootchk

# FIN DEL ARCHIVO makefile

# Ejecución del mismo kernel utilizando otro bootloader

Código del archivo loader\_mtx.s

! This is a very basic example of bootlader for a tiny operating system.

! This is an os loader only!

! It can be loaded at the first sector of a floppy disk:

! cylinder: 0

! sector: 1

! head: 0

! The code in this file is supposed to load

! the kernel (mtx) and to pass control over it.

! The kernel code should be on floppy at:

! cylinder: 0

! sector: 2

! head: 0

! loader file produced by this code should be less or

! equal to 512 bytes, since this is the size of the boot sector.

! boot record is loaded at 0000:7c00h by BIOS

!org 7c00h

!.globl \_print\_string

BOOTSEG = 0x9800 ! Boot block is loaded again to here.

OSSEG = 0x1000

SSP = 32\*1024

BSECTORS = 1 ! load 18 sectors initially

# https://hackaday.io/project/18868-improbable-avr-8088-substitution-for-pcxt/log/51763-first-attempt-at-raw-808688-assembling

# https://www.win.tue.nl/~aeb/linux/lk/lk-3.html

# "Without the export line, ld86 will complain ld86: no start symbol."

export \_main

\_main:

!! initialize the stack:

!mov ax, #0x7c0

!mov ss, ax

!mov sp, #0x3fe ! top of the stack.

!! set data segment:

!xor ax, ax

!mov ds, ax

mov ax,#BOOTSEG

mov es,ax ! set ES to BOOTSEG=0x9800

xor bx,bx

xor dx,dx

xor cx,cx

incb cl

movb ah,#2

movb al,#BSECTORS

int 0x13

jmpi next,BOOTSEG

next:

mov ax,cs

mov ds,ax

mov ss,ax

mov sp,#SSP

! set default video mode 80x25:

mov ah, #0x00

!mov al, #0x03 ! To use only white color for text (for emu8086)

mov al, #0x12 ! In order to use colors for text (for qemu-system-i386)

int 0x10

! print welcome message:

!lea si, msg

mov si, #msg

call \_print\_string

!===================================

! load the kernel at 1000h:0000h

! 10 sectors starting at:

! cylinder: 0

! sector: 2

! head: 0

! BIOS passes drive number in dl,

! so it's not changed:

mov ah, #0x02 ! read function.

mov al, #5; <-- UPDATE THIS TO THE PROPER VALUE for each

; 16-bit MTX version. 2020.10.09.

mov ch, #0 ; cylinder.

mov cl, #2 ; sector.

mov dh, #0 ; head.

! dl not changed! - drive number.

! es:bx points to receiving

! data buffer:

mov bx, #0x1000

mov es, bx

mov bx, #0

! read!

int 0x13

!===================================

! pass control to kernel:

!jmp 1000h:0000h

!jmpi #0x0000,#0x1000

jmpi 0x0000,OSSEG

!===========================================

!print\_string proc near

\_print\_string:

push ax ! store registers...

push si !

next\_char:

mov al, [si]

cmp al, #0

jz printed

inc si

mov ah, #0x0e ! teletype function.

mov bx, #0x000c ! bL = cyan color

int 0x10

jmp next\_char

printed:

pop si ! re-store registers...

pop ax !

ret

!print\_string endp

msg: .asciz "Loading..."

## Construcción del bootloader loader

make loader

## Copiar el bootloader loader

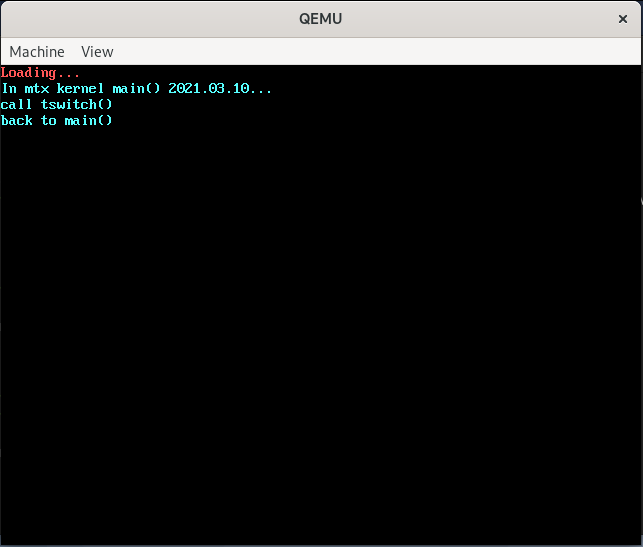
dd if=loader of=mFDimage bs=512 count=1 conv=notrunc

## Copiar el kernel

dd if=mtx of=mfDimage bs=512 seek=1 conv=notrunc

## Correr el kernel usando el bootloader loader

qemu-system-i386 -fda mFDimage -no-fd-bootchk



El objetivo loader en el archivo makefile es como sigue:

loader:loader\_mtx.s

@echo "compiling loader\_mtx.s ..."

as86 -o loader\_mtx.o loader\_mtx.s

@echo "linking loader ....."

ld86 -d -o $@ loader\_mtx.o