LewDos 1.1
Analogic Universal Operating system
Reference Manual

Mutz03 Zockt *

July 2023

*Mutz03 Zockt AKA. Tobias Mittermeier

Contents

1. General Overview

1.1 Modularity

LewDos is a designed to be modular operating system for the purpose of running on a wide variety of Computer systems. The minimal required configuration only requires user input and output. in short a Serial port or paralell port terminal or screen and keyboard. If you wish for more functionality, it is possible to enhance the LewDos operating system with LewDos-modules (.ldm files). Software written for LewDos will work on any LewDos instance, given that the required modules are installed. LewDos is completely virtualiyed and compiles the software bevore runtime for the hardware that its executed from, thus providing maximum compatibility. for example a software that runns on the LewDos-Core System, will run on any LewDos system.

1.2 Kernel and Core system

The LewDos Kernel is written in C, and provides a Virtualisation layer for memory access, athrimetric and User handeling. The Core system is compiled from three parts.
- "Meth" which provides athrimetric and Logic Functionality and also handeles CPU access.
- "Memes" which handles Memory and Stack access, it also provides security integration with "Orgy" to prevent Users to invade each others memory space.
- "butTerm" which is a Terminal access. depending on the Target system it emulates a Terminal on a Graphics screen or it accesses the Serial or Paralell port to output to a terminal.

1.3 Concept

LewDos is a operating system. but at its core, it is a translator to make universal code avaiable. Essentially, there is no compiling or need to compile for LewDos code. the source code will always get shared, and will be compiled at the momment when you run the program. This ensures that the code will run on anything. if you are concerned and do not want to open source every-

thing, there might be a future project to implement some sort of encryption, so only the OS can read the assembly source. that is not priority though. LewDos also offers a simple User interface with "Smash" a instantly interpreted command language, like the Unix Shell, Bash or Basic. LewDos will be Open source and will be avaiable for everyone to compile for themselves. with that we can ensure that Lewdos can be made compatible for every system. if you build your own Homebrew computer. you should be able to Run LewDos on it. LewDos offers a way to access hardware via its own hardware access routines. such enabling you to write modules on OS level to enable certain functionality of your Hardware. Every software program written for LewDos has a header, where all the neccesary modules get specified. so the OS will check if the modules are installed and will promt you with an error if modules cant be found, such preventing unexpected behaviour.

1.4 Module Catalog

on GitHub you will be able to get a place where you can download all the modules. and they will be indexed there as well.

2. Installation of LewDos

3. Usage of Smash

4. LewDos assmebly

4.1 Instruction Set

OPCode (Hex)	Instruction	Function Description	Parameters
0x00	TRP	Trap if instruction memory is empty, Paniks kernel	-
0x01	NOP	No-Operation, does nothing	-
0x02	EXIT	Ends the Program and Exits the Task	-
0x03	PNM	Paniks kernel with custom message; exits program	Message
0x04	HLT	Halts Task and waits until interrupt.	-
0x05	ADD	Add of two operands	A, B and Destination
0x06	SUB	Subtraction A - B	A, B and Destination
0x07	MUL	Multiplication of A and B	A, B and Destination
0x08	DIV	Division A / B	A, B and Destination
0x09	AND	Bitwise A and B	A, B and Destination
0x0A	OR	Bitwise A or B	A, B and Destination
0x0B	XOR	Bitwise A xor B	A, B and Destination
0x0C	NOT	Bitwise Inversion	Source and Destination
0x0D	MOV	Move Values, or load constants	Source and Destination
0x0E	JMP	Jump	Routine Tag
0x0F	SNC	Skip Next Line if condition is met	condition
0x10	SNN	Skip Next Line if condition is not met	condition
0x11	CALL	Push Program status on Stack and jump to Subroutine	Subroutine Tag
0x12	RET	Returns back from subroutine, restors Programn status bevore Call	-
0x13	SINT	Create Subroutine that is triggerable Via Interrupt	Routine Tag
0x14	INT	Trigger Software Interrupt, disables interrupts while routine	SINT Tag
0x15	IRET	returns from interrupt subroutine, reenables interrupts	condition
0x16	AAA	Converts Ascii character into binary value.	Source and Destination
0x17	POP	takes out top value from stack	Destination
0x18	PUSH	pushes value on top of stack	Destination
0x19	WFB	Halts task until Busy flag turns off	-
0x1A	ROT	Binary Rotate	Source, Destination, Direction
0x1B	SHF	Binary Shifting	Source, Destination, Direction

4.2 Memory usage and adressing

In LewDos adressing is quite simple. there are no registers to worry about, the complete system is memory based. Most instructions accept both immidate values and pointers. The memory manager in LewDos "Memes" will allocate a chunk of memory to every user and task on startup and expand if neccesary. those chunks are customizable, factory preset is 512Bytes seperately from the Heap memory avaiable for the user, there is also a Stack, again, dynamicly allocated in chunks of 128Bytes. also customizable. the stack is seperate from the Heap and is not accessable from the user outside of Push and Pop instructions to prevent stack corruption.

4.3 datatypes

LewDos Support many formats of numberic inputs for programming.

Prefix	Type
none	Decimal
$	Hexadecimal
%	Binary
#	Octal

if you need to access memory you need to do it with the character '@' it signs the runtime environment that you are refering to a memory location and not inputting a imidiate value. those pointers are required for for example Destinations but supported for most input fields.

4.4 program flow control

Jumping and Subroutines are supported by LewDos, you can name your routine with a Tag. Write the nameTag of your routine on the top most empty line and the runtime routine will accept it as a Routinetag. while Subroutine Tags and Routine Tags are handled differently in their respective Instructions, they are stored in the same array, so you may not use duplicates of Routine or subroutine tags. note that the tags are case sensitive.

4.5 Stack and Subroutines

LewDos Creates a Stack per user which is dinamicly resized in chunks of 128Bytes. this is reconfigurable. The stack will be avaiable for the User, but will also be used by the Program. When called to a Subroutine, the stack will contain the Program counter of the last remained step of the call. so you want to make sure that you will pop all your values out of the stack bevore returning from a subroutine. the same thing applies for interrupt handeling.

4.6 athrimetric and Logic

The Logic and athrimetric Unit in LewDos supports simple Math and Logic functionality. the math is Integer only, and does not support Floating point. there are Modules that add this functionality though.

5. Architecture

6. Standart modules