CmOS

2025

1 Introduction

In this file, the main concepts used to build CmOS (C modeled OS) will be detailed. The objective of this report is to complete the content of the comments that can be found in the code.

The goal of this project is to understand and apply the main principles of an OS without having to struggle with time demanding parts such as assembly code, bootloader, drivers and so on. Because it is a simulation of an OS done in C, it suits better to the definition of an hypervisor with a single virtual machine than to the one of an operating system.

2 Storage

2.1 Generalities

The storage of CmOS is a file located in bin/disk with a parametric size. To interface with it, src/disk.c gives functions to initialize, write and read it.

3 Programs

3.1 Generalities

For a classic OS, this step would not have been necessary because the programs are written in machine code and that is what is then run by the computer. However, the choice of building a dedicated simple low-level language has been made such that small programs can run on CmOS.

To begin with it, each instruction will consists of an instruction code followed by up to 3 arguments. The RAM size that was set at the beginning was 256 bytes, which leads to an 8-bit architecture so any pointer can be stored.

3.2 CPU

The CPU will have 16 bytes of registers. This way, a register can be accessed with a 4-bit identifiers. This allows any instruction to only take a maximum of 16 bits of arguments. These 16 bits can be up to 4 registers, 2 registers and an immediate byte value or an immediate 2 bytes value (disk address for example).

register name	address	size	use
FLAGS	0x0	8 bits	to be determined
R1	0x1	8 bits	general purpose register
R2	0x2	8 bits	general purpose register
R3	0x3	8 bits	general purpose register
R4	0x4	8 bits	general purpose register
R5	0x5	8 bits	general purpose register
R16	0x6	16 bits	16 bit register for arithmetic
RL	0x6	8 bits	general purpose register
RH	0x7	8 bits	general purpose register
RSI	0x8	16 bits	source pointer
RDI	0xA	16 bits	destination pointer
RI	0xC	16 bits	instruction pointer
RS	0xE	16 bits	stack pointer

Table 1: CPU registers

CF	ZF	SF	OF	*	*	*	*
Carry Flag	Zero Flag	Sign Flag	Overflow Flag				

Table 2: FLAGS register

3.3 Instruction set

3.3.1 8-bit configuration

In the instruction table, the following convention is followed:

- r* refers to a register (4 bits)
- imm* refers to an immediate value
- mem refers to a memory address (8 bits)
- \bullet \star indicates an unused field
- ullet an italic field indicates an input
- a **bold** field indicates an output

$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	Instruction	Op Code	byte1	byte2	16 bit reg compatibility	comment			
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	AND	0x00	r1 r2	* r3		$\mathbf{r3} = r1 \text{ AND } r2$			
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	AND	0x01	<i>r1</i> r2	imm8		$\mathbf{r2} = r1 \; \mathrm{AND} \; imm8$			
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	OP	0x02	r1 r2	* r3		$\mathbf{r3} = r1 \text{ OR } r2$			
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	OK	0x03	<i>r</i> 1 r 2	imm8		$\mathbf{r2} = r1 \mathrm{OR} imm8$			
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	NOT	0x04	<i>r1</i> r2	*		$\mathbf{r2} = \mathrm{NOT} \ r1$			
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	CIII	0x04	r1 r2	* r3	•	$\mathbf{r3} = r1 \ll r2$			
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	SIL	0x05	<i>r</i> 1 r 2	imm8	•	$\mathbf{r2} = r1 \ll imm8$			
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	СПР	0x06	r1 r2	* r3	•	$\mathbf{r3}=r1\gg r2$			
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	SIIK	0x07	<i>r</i> 1 r 2	imm8	•	$\mathbf{r2}=r1\gg imm8$			
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	ADD	0x10	r1 r2	* r3	•	$\mathbf{r3} = r1 + r2$			
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	ADD	0x11	<i>r</i> 1 r 2	imm8	•	$\mathbf{r2}=\mathit{r1}+\mathit{imm8}$			
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$		0x12	r1 r2		•	$\mathbf{r3} = r1 - r2$			
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	SUB	0x13	<i>r</i> 1 r 2	imm8	•	$\mathbf{r2} = r1 - imm8$			
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$		0x14	imm8	<i>r1</i> r2		$\mathbf{r2}=\mathit{imm8}$ - $\mathit{r1}$			
	MILI	0x15	r1 r2	*	•	$\mathbf{R16} = r1 * r2$			
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	MUL	0x16	r1 *	imm8	•	$\mathbf{R16} = r1 * imm8$			
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	IDIV	0x17	<i>r</i> 1 r 2	*	•	$\mathbf{r2}=R16\;//\;r1$			
MOD $0x1A$ $imm8$ * $\mathbf{r1}$ • $\mathbf{r1} = R16 \text{ mod } imm8$ MOV $0x20$ $r1$ $\mathbf{r2}$ * • $\mathbf{r2} = r1$ $0x21$ * $\mathbf{r1}$ $imm8$ • $\mathbf{R16} = imm8$ $0x22$ $imm16$ • $\mathbf{R16} = imm16$ LOAD $0x23$ * $\mathbf{r1}$ mem * • $\mathbf{R16} = *(mem)$ STORE $0x24$ mem * • $\mathbf{R16} = *(mem)$ STORE $0x25$ * $r1$ mem * • $raises$ \mathbf{ZF} , $simm = r1$ CMP $0x30$ $r1$ $r2$ * • $raises$ \mathbf{ZF} , $simm = r1$ TEST $0x31$ $r1$ $r2$ * • $raises$ \mathbf{ZF} , $simm = r2$ SKIFZ $0x40$ * * * $simm = r2$ SKIFNZ $0x40$ * * * $simm = r2$ PRNT $0x40$ * * * $simm = r2$ $simm $	IDIV	0x18	imm8	* r1	•	${f r1}=R16\;//\;imm8$			
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	MOD	0x19	<i>r</i> 1 r 2	*	•	$\mathbf{r2} = R16 \mod r1$			
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	MOD	0x1A	imm8	* r1	•	$\mathbf{r1} = R16 \mod imm8$			
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$		0x20	<i>r1</i> r2	*	•	$\mathbf{r2} = r1$			
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	MOV	0x21	* r1	imm8		$\mathbf{r1}=\mathit{imm8}$			
BOAD 0x24 mem * • R16 = *(mem) STORE 0x25 * r1 mem *(mem) = r1 0x26 mem * • raises ZF, SF for (r2 - r1) CMP 0x30 r1 r2 * raises ZF for (r2 - r1) TEST 0x31 r1 r2 * r1 AND 0b1 « r2 0x32 r1 * imm8 * skip next instruction if ZF = 1 SKIFZ 0x40 * * * skip next instruction if ZF = 0 PRNT 0xF0 mem * r1 • prints r1 characters from mem		0x22	imn	n16	•	${f R16}=imm16$			
STORE 0x24 mem \star • R16 = *(mem) STORE 0x25 \star r1 mem \star • *(mem) = r1 0x26 mem \star • raises ZF , SF for $(r2 - r1)$ raises ZF for $r1$ AND 0b1 $\ll r2$ 0x31 r1 r2 \star raises ZF for $r1$ AND 0b1 $\ll r2$ 0x32 r1 \star imm8 • skip next instruction if $ZF = 1$ SKIFZ 0x40 \star \star \star skip next instruction if $ZF = 1$ SKIFNZ 0x40 \star \star \star prints $r1$ characters from mem	LOAD	0x23	* r1	mem		$\mathbf{r1} = *(mem)$			
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	LOAD	0x24	mem	*	•				
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	STODE	0x25	* r1	mem					
TEST $ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	SIORE	0x26	mem	*	•	$*(\mathbf{mem}) = R16$			
TEST	CMP	0x30	r1 r2	*	•	raises ZF , SF for $(r2 - r1)$			
TEST $0x32 r1 \star imm8 \qquad \bullet \qquad \begin{array}{c} r1 \text{ AND 0b1} \ll r2 \\ raises \textbf{ZF for} \\ r1 \text{ AND 0b1} \ll imm8 \\ \hline \\ SKIFZ \\ SKIFZ \\ \hline \\ 0x40 \\ \hline \\ \\ SKIFNZ \\ \hline \\ 0x40 \\ \hline \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ $		021	10			raises ZF for			
$0x32 \qquad r1 \qquad imm8 \qquad \bullet \qquad raises \textbf{ZF} \text{ for } \\ r1 \text{ AND } 0b1 \ll imm8 \\ \hline SKIFZ \qquad 0x40 \qquad \star \qquad \star \qquad \\ SKIFNZ \qquad 0x40 \qquad \star \qquad \star \qquad \\ SKIFNZ \qquad 0x40 \qquad \star \qquad \star \qquad \star \qquad \\ PRNT \qquad 0xF0 \qquad mem \qquad \star \qquad r1 \qquad \bullet \qquad prints \ r1 \text{ characters from } mem \\ \hline $		UX31	TI TZ	*		$r1 \text{ AND } 0b1 \ll r2$			
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$		0,,29	m1 1	imm 0		raises ZF for			
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$		UX32	T1 *	ummo	•	$r1 \text{ AND } 0b1 \ll imm8$			
SKIFNZ $0x40$ \star \star \star Skip next instruction if $ZF=1$ skip next instruction if $ZF=0$ PRNT $0xF0$ mem \star $r1$ \bullet prints $r1$ characters from mem	CIZIEZ	040				skip next instruction			
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	SKIFZ	UX4U	*	*		$\text{if } \mathit{ZF} = 1$			
PRNT $0xF0$ mem \star $r1$ \bullet prints $r1$ characters from mem	CKIENZ	0×40	-1			skip next instruction			
1	SIXITIVA	UX40	*	*		$\text{if } \mathit{ZF} = 0$			
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	PRNT	0xF0	mem	⋆ r1	•	prints r1 characters from mem			
	HALT	0xFF	*	*		stops the CPU			

Table 3: 8-bit instruction set