Euro One Manual

Maximilian Noppel, max@noppelmax.online

7. Februar 2020

Inhaltsverzeichnis

1 Introduction

2 Features

- 8-Bit Data and 16-Bit Addressbus
- 4x General Purpose 8-Bit-Registers (Could run as 2x 16-Bit-Registers)
- 64kB of RAM
- StackPointer Register
- BasePointer Register
- ProgramCounter Register
- Peripherie: TTY Interface, 8x8 LED-Matrix with double framebuffer, 1x 16-Bit-Timer, 1x 8-Bit-GPIO
- Conditional Jumps: `jz`,`jnz`,`jg`,`jge`,`jl`,`jle`,`je`
- Stackframeoperations: `call`,`ret`,`pop`,`push`
- Dynamic Segmented Adressing: `storerrao`, `loadraor`
- Microprogramming by DiodeMatrix
- 16-Bit muliplication result
- 16-Bit division input
- Arithmetics: `add`,`sub`,`imul`,`idiv`,`and`,`or`,`xor`,`not`,`neg`

3 Logic

3.1 Registers

Shortname	Name	Type	Id
r00	Constant Zero Register	for internal use	`0x00`
r01	Constant One Register	for internal use	`0x01`
r02	Constant Two Register	for internal use	`0x02`
r03	Constant Three Register	for internal use	`0x03`
ra	8-Bit General Purpose Register A	General Purpose Register	`0x04`
rb	8-Bit General Purpose Register B	General Purpose Register	`0x05`
rc	8-Bit General Purpose Register C	General Purpose Register	`0x06`
rd	8-Bit General Purpose Register D	General Purpose Register	`0x07`
rpc	16-Bit Program Counter	Spezial Purpose Register	`0x98`
rsp	16-Bit Stack Pointer	Spezial Purpose Register	`0xBA`
rbp	16-Bit Base Pointer	Spezial Purpose Register	`0xDC`
rtty	8-Bit TTY Console	Spezial Purpose Register	`0x0E`
rop0	8-Bit Operator Register	for internal use	`0x18`
rop1	8-Bit Operator Register	for internal use	`0x19`
rop2	8-Bit Operator Register	for internal use	`0x1A`
rab	16-Bit General Purpose Register (AB)	General Purpose Register, overlapping with ra,rb	`0x45`
rbc	16-Bit General Purpose Register (BC)	General Purpose Register, overlapping with rb,rc	`0x56`
rcd	16-Bit General Purpose Register (CD)	General Purpose Register, overlapping with rc,rd	`0x67`
rda	16-Bit General Purpose Register (DA)	General Purpose Register, overlapping with ra,rd	`0x74`
rba	16-Bit General Purpose Register (BA)	General Purpose Register, overlapping with ra,rb	`0x54`
rcb	16-Bit General Purpose Register (CB)	General Purpose Register, overlapping with rb,rc	`0x65`
rdc	16-Bit General Purpose Register (DC)	General Purpose Register, overlapping with rc,rd	`0x76`
rad	16-Bit General Purpose Register (AD)	General Purpose Register, overlapping with ra,rd	`0x47`

3.2 ALU

Bit	Name	Function
0	`CF`	Carry Flag

1	ZF	Zero Flag
2	SF	Sign Flag
3	OF	Overflow Flag

3.3 Control Unit

3.4 Microprogramming

3.5 Selectors

Databus and addressbus, ALU Ins and CarryIn, Store	SelectCode
const 00	0x00
const 01	0x01
const 02	0x02
const 03	0x03
A	0x04
B	0x05
C	0x06
D	0x07
PC_low	0x08
PC_high	0x09
SP_low	0x0A
SP_high	0x0B
BP_low	0x0C
BP_hight	0x0D
CARRY	0x0E
unused	0x0F
reg0l	0x10
reg0h	0x11
reg1l	0x12
reg1h	0x13
reg2l	0x14
reg2h	0x15
unused	0x16
unused	0x17
op0	0x18
op1	0x19
op2	0x1A
RAM	0x1B
alu_result	0x1C
alu_mulcarry	0x1D
alu_flags	0x1E
FLAGS	0x1F

3.5.1 Signals

- REG0_TO_DATABUS
- REG0_TO_ADDRESSBUS
- STORE_TO_REG0

4 Mechnical aspects

5 Hardware aspects

5.1 Order and assembly

6 Machinecode

6.1 Commands: dataflow

Commandname	op0	op1	op2	Description	Binary
MOVRR	reg8	reg8		Copies the value of first 8-bit register to the second.	0x01
MOVCR	const8	reg8		Writes a 8-bit-constant 8-bit value in the given register	0x02
MOVCR16	const16h	const16l	reg16	Writes a 16-bit constant to a 16-bit register	0x80
LOADAR	addr16h	addr16l	reg8		0x03
LOADRAR	reg16	reg8			0x04
LOADRAOR	reg16	const8	reg8		0x05
STORERA	reg8	addr16h	addr16l		0x06
STORERRA	reg8	reg16			0x07
STORERRAO	reg8	reg16	const8		0x08
STORECA	const8	addr16h	addr16l		0x47
STORECRA	const8	reg16			0x48
STORECRAO	const8	reg16	const8		0x49

6.2 Commands: datamanipulation

Commandname	op0	op1	op2	Description	Binary
RCL	reg8				0x19
RCR	reg8				0x1A
ROL	reg8				0x1B
ROR	reg8				0x1C
SAL	reg8				0x1D
SAR	reg8				0x1E
SHL	reg8				0x1F
SHR	reg8				0x20
AND	reg8	reg8			0x30
AND16	reg16	reg16			0xAA
ANDC	reg8	const8			0x31
ANDC16	reg16	const16h	const16l		0xAB
OR	reg8	reg8			0x32
OR16	reg16	reg16			0xAC
ORC	reg8	const8			0x33
ORC16	reg16	const16h	const16l		0xAD
XOR	reg8	reg8			0x34
XORC	reg8	const8			0x35
NEG	reg8	reg8			0x36
NEGC	reg8	const8			0x37
NOT	reg8	reg8			0x38
NOTC	reg8	const8			0x39

6.3 Commands: controlflow

Commandname	op0	op1	op2	Description	Binary
INT					0x8D
EINT					0x8E
JMP	addr16h	addr16l			0x0F
JRA	reg16				0x10

JZ	addr16h	addr16l	0x11
JNZ	addr16h	addr16l	0x12
JE	addr16h	addr16l	0x13
JNE	addr16h	addr16l	0x14
JG	addr16h	addr16l	0x15
JGE	addr16h	addr16l	0x16
JL	addr16h	addr16l	0x17
JLE	addr16h	addr16l	0x18

6.4 Commands: arithmetics

Commandname	op0	op1	op2	Description	Binary
ADD	reg8	reg8			0x21
ADD16	reg16	reg16			0xA0
ADDC	reg8	const8			0x22
ADDC16	reg16	const16h	const16l		0xA1
SUB	reg8	reg8			0x23
SUB16	reg16	reg16			0xA2
SUBC	reg8	const8			0x24
SUBC16	reg16	const16h	const16l		0xA3
IMUL	reg8	reg8			0x25
IMULC	reg8	const8			0x26
IDIV	reg8	reg8			0x27
IDIVC	reg8	const8			0x28
INC	reg8				0x29
INC16	reg16				0xA8
DEC	reg8				0x2A
DEC16	reg16				0xA9
MOD	reg8	reg8			0x43
MODC	reg8	const8			0x44

6.5 Commands: stackhandling

Commandname	op0	op1	op2	Description	Binary
PUSH	reg8				0x09
POP	reg8				0x0A
CALL	addr16h	addr16l			0x0B
RET					0x0C

6.6 Commands: spezial

Commandname	op0	op1	op2	Description	Binary
HLT				Stops the clock. Can only be reset be manually!	0x00
RST					0x41
NOP					0x42
CMP	reg8	reg8			0x45
CMPC	reg8	const8			0x46

6.7 HLT

Usage: `HLT`
Suggested assembler command: `HLT`
Short description: Stops the clock. Can only be reset be manually!

6.8 MOVRR

Usage: `MOVRR reg8 reg8`
Suggested assembler command: `MOV`
Short description: Copies the value of first 8-bit register to the second.

6.9 MOVCR

Usage: `MOVCR const8 reg8`
Suggested assembler command: `MOV`
Short description: Writes a 8-bit-constant 8-bit value in the given register

6.10 MOVCR16

Usage: `MOVCR16 const16h const16l reg16`
Suggested assembler command: `MOV`
Short description: Writes a 16-bit constant to a 16-bit register

6.11 LOADAR

Usage: `LOADAR addr16h addr16l reg8`
Suggested assembler command: `LOAD`
Short description:

6.12 LOADRAR

Usage: `LOADRAR reg16 reg8`
Suggested assembler command: `LOAD`
Short description:

6.13 LOADRAOR

Usage: `LOADRAOR reg16 const8 reg8`
Suggested assembler command: `LOAD`
Short description:

6.14 STORERA

Usage: `STORERA reg8 addr16h addr16l`
Suggested assembler command: `STORE`
Short description:

6.15 STORERRA

Usage: `STORERRA reg8 reg16`
Suggested assembler command: `STORE`
Short description:

6.16 STORERRAO

Usage: `STORERRAO reg8 reg16 const8`
Suggested assembler command: `STORE`
Short description:

6.17 PUSH

Usage: `PUSH reg8`
Suggested assembler command: `PUSH`
Short description:

6.18 POP

Usage: `POP reg8`
Suggested assembler command: `POP`
Short description:

6.19 CALL

Usage: `CALL addr16h addr16l`
Suggested assembler command: `CALL`
Short description:

6.20 RET

Usage: `RET`
Suggested assembler command: `RET`
Short description:

6.21 INT

Usage: `INT`
Suggested assembler command: `INT`
Short description:

6.22 EINT

Usage: `EINT`
Suggested assembler command: `EINT`
Short description:

6.23 JMP

Usage: `JMP addr16h addr16l`
Suggested assembler command: `JMP`
Short description:

6.24 JRA

Usage: `JRA reg16`
Suggested assembler command: `JMP`
Short description:

6.25 JZ

Usage: `JZ addr16h addr16l`
Suggested assembler command: `JZ`
Short description:

6.26 JNZ

Usage: `JNZ addr16h addr16l`
Suggested assembler command: `JNZ`
Short description:

6.27 JE

Usage: `JE addr16h addr16l`
Suggested assembler command: `JE`
Short description:

6.28 JNE

Usage: `JNE addr16h addr16l`
Suggested assembler command: `JNE`
Short description:

6.29 JG

Usage: `JG addr16h addr16l`
Suggested assembler command: `JG`
Short description:

6.30 JGE

Usage: `JGE addr16h addr16l`
Suggested assembler command: `JGE`
Short description:

6.31 JL

Usage: `JL addr16h addr16l`
Suggested assembler command: `JL`
Short description:

6.32 JLE

Usage: `JLE addr16h addr16l`
Suggested assembler command: `JLE`
Short description:

6.33 RCL

Usage: `RCL reg8`
Suggested assembler command: `RCL`
Short description:

6.34 RCR

Usage: `RCR reg8`
Suggested assembler command: `RCR`
Short description:

6.35 ROL

Usage: `ROL reg8`
Suggested assembler command: `ROL`
Short description:

6.36 ROR

Usage: `ROR reg8`
Suggested assembler command: `ROR`
Short description:

6.37 SAL

Usage: `SAL reg8`
Suggested assembler command: `SAL`
Short description:

6.38 SAR

Usage: `SAR reg8`
Suggested assembler command: `SAR`
Short description:

6.39 SHL

Usage: `SHL reg8`
Suggested assembler command: `SHL`
Short description:

6.40 SHR

Usage: `SHR reg8`
Suggested assembler command: `SHR`
Short description:

6.41 ADD

Usage: `ADD reg8 reg8`
Suggested assembler command: `ADD`
Short description:

6.42 ADD16

Usage: `ADD16 reg16 reg16`
Suggested assembler command: `ADD`
Short description:

6.43 ADDC

Usage: `ADDC reg8 const8`
Suggested assembler command: `ADD`
Short description:

6.44 ADDC16

Usage: `ADDC16 reg16 const16h const16l`
Suggested assembler command: `ADD`
Short description:

6.45 SUB

Usage: `SUB reg8 reg8`
Suggested assembler command: `SUB`
Short description:

6.46 SUB16

Usage: `SUB16 reg16 reg16`
Suggested assembler command: `SUB`
Short description:

6.47 SUBC

Usage: `SUBC reg8 const8`
Suggested assembler command: `SUB`
Short description:

6.48 SUBC16

Usage: `SUBC16 reg16 const16h const16l`
Suggested assembler command: `SUB`
Short description:

6.49 IMUL

Usage: `IMUL reg8 reg8`
Suggested assembler command: `IMUL`
Short description:

6.50 IMULC

Usage: `IMULC reg8 const8`
Suggested assembler command: `IMUL`
Short description:

6.51 IDIV

Usage: `IDIV reg8 reg8`
Suggested assembler command: `IDIV`
Short description:

6.52 IDIVC

Usage: `IDIVC reg8 const8`
Suggested assembler command: `IDIV`
Short description:

6.53 INC

Usage: `INC reg8`
Suggested assembler command: `INC`
Short description:

6.54 INC16

Usage: `INC16 reg16`
Suggested assembler command: `INC`
Short description:

6.55 DEC

Usage: `DEC reg8`
Suggested assembler command: `DEC`
Short description:

6.56 DEC16

Usage: `DEC16 reg16`
Suggested assembler command: `DEC`
Short description:

6.57 AND

Usage: `AND reg8 reg8`
Suggested assembler command: `AND`
Short description:

6.58 AND16

Usage: `AND16 reg16 reg16`
Suggested assembler command: `AND`
Short description:

6.59 ANDC

Usage: `ANDC reg8 const8`
Suggested assembler command: `AND`
Short description:

6.60 ANDC16

Usage: `ANDC16 reg16 const16h const16l`
Suggested assembler command: `AND`
Short description:

6.61 OR

Usage: `OR reg8 reg8`
Suggested assembler command: `OR`
Short description:

6.62 OR16

Usage: `OR16 reg16 reg16`
Suggested assembler command: `OR`
Short description:

6.63 ORC

Usage: `ORC reg8 const8`
Suggested assembler command: `OR`
Short description:

6.64 ORC16

Usage: `ORC16 reg16 const16h const16l`
Suggested assembler command: `OR`
Short description:

6.65 XOR

Usage: `XOR reg8 reg8`
Suggested assembler command: `XOR`
Short description:

6.66 XORC

Usage: `XORC reg8 const8`
Suggested assembler command: `XOR`
Short description:

6.67 NEG

Usage: `NEG reg8 reg8`
Suggested assembler command: `NEG`
Short description:

6.68 NEGC

Usage: `NEGC reg8 const8`
Suggested assembler command: `NEG`
Short description:

6.69 NOT

Usage: `NOT reg8 reg8`
Suggested assembler command: `NOT`
Short description:

6.70 NOTC

Usage: `NOTC reg8 const8`
Suggested assembler command: `NOT`
Short description:

6.71 RST

Usage: `RST`
Suggested assembler command: `RST`
Short description:

6.72 NOP

Usage: `NOP`
Suggested assembler command: `NOP`
Short description:

6.73 MOD

Usage: `MOD reg8 reg8`
Suggested assembler command: `MOD`
Short description:

6.74 MODC

Usage: `MODC reg8 const8`
Suggested assembler command: `MOD`
Short description:

6.75 CMP

Usage: `CMP reg8 reg8`
Suggested assembler command: `CMP`
Short description:

6.76 CMPC

Usage: `CMPC reg8 const8`
Suggested assembler command: `CMP`
Short description:

6.77 STORECA

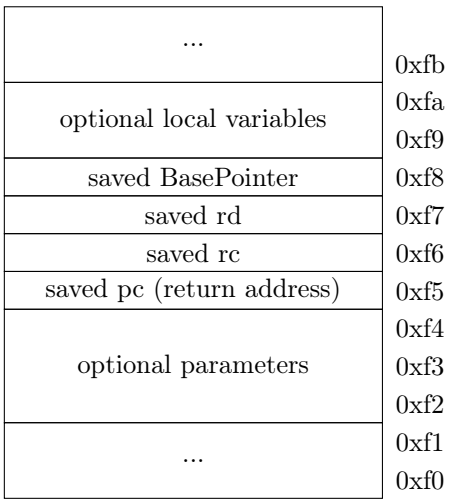
Usage: `STORECA const8 addr16h addr16l`
Suggested assembler command: `STORE`
Short description:

6.78 STORECRA

Usage: `STORECRA const8 reg16`
Suggested assembler command: `STORE`
Short description:

6.79 STORECRAO

Usage: `STORECRAO const8 reg16 const8`
Suggested assembler command: `STORE`
Short description:



6.80 Accessing the memory

There are a bunch of options to access a bigger memoryspace as the wordwidth indicates. In my case the Addressbus is 16-bit width, while the Databus only got 8-bit.

The possible options i thought about were:

- using multiple register for addressing
- using one special 16-bit register for addressing
- using one additional 8-bit register for the upper 8-bits of a address
- using some kind of memory banking

[1]

6.81 Segmented Adressing

[2]

6.82 Framehandling

Bei einem Aufruf von `call` werden die Register `rc` und `rb` gesichert und beim Verlassen des Stackframes wiederhergestellt. Es wird empfohlen die beiden Register `ra` und `rb` zur Übergabe von Parametern zu verwenden. Ansonsten können Parameter natürlich wie üblich über den RAM übergeben werden. Zur Offsetberechnung ist folgender Stackframeaufbau zu beachten:

7 Assembler

7.1 Register

7.2 Constants

7.3 Adresses

7.4 Labels

7.5 Variables

7.6 Commands

7.7 Comments

8 Peripherie

The Logisim-Version uses a simple TTY-Display to print out text. The peripherie addresses start at `0xE0` which is the TTY.

Registername	Description	Address
`TTY_OUT`	Every write to this address will appear as ASCII character on the ttyscreen	`0xE0`
`TIMER0_OPT`		`0xE1`
`TIMER0_LOW`		`0xE2`
`TIMER0_HIGH`		`0xE3`
`NONE`		`0xE4`
`NONE`		`0xE5`
`NONE`		`0xE6`
`NONE`		`0xE7`
`NONE`		`0xE8`
`NONE`		`0xE9`
`NONE`		`0xEA`
`NONE`		`0xEB`
`NONE`		`0xEC`
`NONE`		`0xED`
`NONE`		`0xEE`
`SELECT_FB`	Bit0: 0->FB0, 1->FB1	`0xEF`
`FB0_0`		`0xF0`
`FB0_1`		`0xF1`
`FB0_2`		`0xF2`
`FB0_3`		`0xF3`
`FB0_4`		`0xF4`
`FB0_5`		`0xF5`
`FB0_6`		`0xF6`
`FB0_7`		`0xF7`
`FB1_0`		`0xF8`
`FB1_1`		`0xF9`
`FB1_2`		`0xFA`
`FB1_3`		`0xFB`
`FB1_4`		`0xFC`
`FB1_5`		`0xFD`
`FB1_6`		`0xFE`
`FB1_7`		`0xFF`

8.1 TTY_OUT

8.2 TIMER0_OPT

8.3 TIMER0_LOW

8.4 TIMER0_HIGH

8.5 SELECT_FB

8.6 FBx_y

9 EuroLang

10 Compiler

11 TODO

- DebuggingInterface / Breakpoints
- 16bit Adressbus
- 24bit Adressbus/16 bit virtual
- 16-bit Operations
- Timer implementieren
- Interrupts /ISR
- Adressoffset, static, dynamic

Literatur

[1] phuclv, "How can 8-bit processor support more than 256 bytes of ram?."

[2] "x86 assembly language - wikipedia."