VU Abstrakte Maschinen
i8080.js

Intel 8080
implementiert in
JavaScript/TypeScript

Moritz Bauer 0649647

Juni 23, 2020

Ziele

- Intel 8080 implementieren in JS/TS
- Alle Operationen (ohne IN,OUT, EI, DI, RST)
- Simpler Assambler um Programme zu schreiben
- UI in JS (läuft vollständig im Browser)

Specs

- 8 Allgemeine Register (A B C D E H L M)
  - A dient als Akkumulator
  - H L dienen zur Memory-Adressierung
  - M referenziert auf Memory (durch HL adressiert)
- 16 Bit Register Paare (**B**C, **D**E, **H**L, **PSW***)
- 16 Bit Stack Pointer und Programm Counter Register
- 5 x 1Bit Flag Regeister (Carry, Sign, Zero, Parity, Auxiliary)
- 16 Bit Memory Adressen → 64 KB Memory adressieren
- 16 Bit Operationen/ALU (INX,DCX, DAD, LXI, PUSH POP)
- Support für Dezimal Addition

*Program Status Word

OPCODE

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	NOP 1 4 - - - - -	LXI B,d16 3 10 - - - - -	STAX B 1 7 - - - - -	INX B 1 5 - - - - -	INR B 1 5 S Z A P -	DCR B 1 5 S Z A P -	MVI B,d8 2 7 - - - - -	RLC 1 4 - - - - C	*NOP 1 4 - - - - -	DAD B 1 10 - - - - C	LDAX B 1 7 - - - - -	DCX B 1 5 - - - - -	INR C 1 5 S Z A P -	DCR C 1 5 S Z A P -	MVI C,d8 2 7 - - - - -	RRC 1 4 - - - - C
1x	*NOP 1 4 - - - - -	LXI D,d16 3 10 - - - - -	STAX D 1 7 - - - - -	INX D 1 5 - - - - -	INR D 1 5 S Z A P -	DCR D 1 5 S Z A P -	MVI D,d8 2 7 - - - - -	RAL 1 4 - - - - C	*NOP 1 4 - - - - -	DAD D 1 10 - - - - C	LDAX D 1 7 - - - - -	DCX D 1 5 - - - - -	INR E 1 5 S Z A P -	DCR E 1 5 S Z A P -	MVI E,d8 2 7 - - - - -	RAR 1 4 - - - - C
2x	*NOP 1 4 - - - - -	LXI H,d16 3 10 - - - - -	SHLD a16 3 16 - - - - -	INX H 1 5 - - - - -	INR H 1 5 S Z A P -	DCR H 1 5 S Z A P -	MVI H,d8 2 7 - - - - -	DAA 1 4 S Z A P C	*NOP 1 4 - - - - -	DAD H 1 10 - - - - C	LHLD a16 3 16 - - - - -	DCX H 1 5 - - - - -	INR L 1 5 S Z A P -	DCR L 1 5 S Z A P -	MVI L,d8 2 7 - - - - -	CMA 1 4 - - - - -
3x	*NOP 1 4 - - - - -	LXI SP,d16 3 10 - - - - -	STA a16 3 13 - - - - -	INX SP 1 5 - - - - -	INR M 1 10 S Z A P -	DCR M 1 10 S Z A P -	MVI M,d8 2 10 - - - - -	STC 1 4 - - - - C	*NOP 1 4 - - - - -	DAD SP 1 10 - - - - C	LDA a16 3 13 - - - - -	DCX SP 1 5 - - - - -	INR A 1 5 S Z A P -	DCR A 1 5 S Z A P -	MVI A,d8 2 7 - - - - -	CMC 1 4 - - - - C
4x	MOV B,B 1 5 - - - - -	MOV B,C 1 5 - - - - -	MOV B,D 1 5 - - - - -	MOV B,E 1 5 - - - - -	MOV B,H 1 5 - - - - -	MOV B,L 1 5 - - - - -	MOV B,M 1 7 - - - - -	MOV B,A 1 5 - - - - -	MOV C,B 1 5 - - - - -	MOV C,C 1 5 - - - - -	MOV C,D 1 5 - - - - -	MOV C,E 1 5 - - - - -	MOV C,H 1 5 - - - - -	MOV C,L 1 5 - - - - -	MOV C,M 1 7 - - - - -	MOV C,A 1 5 - - - - -
5x	MOV D,B 1 5 - - - - -	MOV D,C 1 5 - - - - -	MOV D,D 1 5 - - - - -	MOV D,E 1 5 - - - - -	MOV D,H 1 5 - - - - -	MOV D,L 1 5 - - - - -	MOV D,M 1 7 - - - - -	MOV D,A 1 5 - - - - -	MOV E,B 1 5 - - - - -	MOV E,C 1 5 - - - - -	MOV E,D 1 5 - - - - -	MOV E,E 1 5 - - - - -	MOV E,H 1 5 - - - - -	MOV E,L 1 5 - - - - -	MOV E,M 1 7 - - - - -	MOV E,A 1 5 - - - - -
6x	MOV H,B 1 5 - - - - -	MOV H,C 1 5 - - - - -	MOV H,D 1 5 - - - - -	MOV H,E 1 5 - - - - -	MOV H,H 1 5 - - - - -	MOV H,L 1 5 - - - - -	MOV H,M 1 7 - - - - -	MOV H,A 1 5 - - - - -	MOV L,B 1 5 - - - - -	MOV L,C 1 5 - - - - -	MOV L,D 1 5 - - - - -	MOV L,E 1 5 - - - - -	MOV L,H 1 5 - - - - -	MOV L,L 1 5 - - - - -	MOV L,M 1 7 - - - - -	MOV L,A 1 5 - - - - -
7x	MOV M,B 1 7 - - - - -	MOV M,C 1 7 - - - - -	MOV M,D 1 7 - - - - -	MOV M,E 1 7 - - - - -	MOV M,H 1 7 - - - - -	MOV M,L 1 7 - - - - -	HLT 1 7 - - - - -	MOV M,A 1 7 - - - - -	MOV A,B 1 5 - - - - -	MOV A,C 1 5 - - - - -	MOV A,D 1 5 - - - - -	MOV A,E 1 5 - - - - -	MOV A,H 1 5 - - - - -	MOV A,L 1 5 - - - - -	MOV A,M 1 7 - - - - -	MOV A,A 1 5 - - - - -
8x	ADD B 1 4 S Z A P C	ADD C 1 4 S Z A P C	ADD D 1 4 S Z A P C	ADD E 1 4 S Z A P C	ADD H 1 4 S Z A P C	ADD L 1 4 S Z A P C	ADD M 1 7 S Z A P C	ADD A 1 4 S Z A P C	ADC B 1 4 S Z A P C	ADC C 1 4 S Z A P C	ADC D 1 4 S Z A P C	ADC E 1 4 S Z A P C	ADC H 1 4 S Z A P C	ADC L 1 4 S Z A P C	ADC M 1 7 S Z A P C	ADC A 1 4 S Z A P C
9x	SUB B 1 4 S Z A P C	SUB C 1 4 S Z A P C	SUB D 1 4 S Z A P C	SUB E 1 4 S Z A P C	SUB H 1 4 S Z A P C	SUB L 1 4 S Z A P C	SUB M 1 7 S Z A P C	SUB A 1 4 S Z A P C	SBB B 1 4 S Z A P C	SBB C 1 4 S Z A P C	SBB D 1 4 S Z A P C	SBB E 1 4 S Z A P C	SBB H 1 4 S Z A P C	SBB L 1 4 S Z A P C	SBB M 1 7 S Z A P C	SBB A 1 4 S Z A P C
Ax	ANA B 1 4 S Z A P C	ANA C 1 4 S Z A P C	ANA D 1 4 S Z A P C	ANA E 1 4 S Z A P C	ANA H 1 4 S Z A P C	ANA L 1 4 S Z A P C	ANA M 1 7 S Z A P C	ANA A 1 4 S Z A P C	XRA B 1 4 S Z A P C	XRA C 1 4 S Z A P C	XRA D 1 4 S Z A P C	XRA E 1 4 S Z A P C	XRA H 1 4 S Z A P C	XRA L 1 4 S Z A P C	XRA M 1 7 S Z A P C	XRA A 1 4 S Z A P C
Bx	ORA B 1 4 S Z A P C	ORA C 1 4 S Z A P C	ORA D 1 4 S Z A P C	ORA E 1 4 S Z A P C	ORA H 1 4 S Z A P C	ORA L 1 4 S Z A P C	ORA M 1 7 S Z A P C	ORA A 1 4 S Z A P C	CMP B 1 4 S Z A P C	CMP C 1 4 S Z A P C	CMP D 1 4 S Z A P C	CMP E 1 4 S Z A P C	CMP H 1 4 S Z A P C	CMP L 1 4 S Z A P C	CMP M 1 7 S Z A P C	CMP A 1 4 S Z A P C
Cx	RNZ 1 11/5 - - - - -	POP B 1 10 - - - - -	JNZ a16 3 10 - - - - -	JMP a16 3 10 - - - - -	CNZ a16 3 17/11 - - - - -	PUSH B 1 11 - - - - -	ADI d8 2 7 S Z A P C		RZ 1 11/5 - - - - -	RET 1 10 - - - - -	JZ a16 3 10 - - - - -	*JMP a16 3 10 - - - - -	CZ a16 3 17/11 - - - - -	CALL a16 3 17 - - - - -	ACI d8 2 7 S Z A P C	
Dx	RNC 1 11/5 - - - - -	POP D 1 10 - - - - -	JNC a16 3 10 - - - - -		CNC a16 3 17/11 - - - - -	PUSH D 1 11 - - - - -	SUI d8 2 7 S Z A P C		RC 1 11/5 - - - - -	*RET 1 10 - - - - -	JC a16 3 10 - - - - -		CC a16 3 17/11 - - - - -	*CALL a16 3 17 - - - - -	SBI d8 2 7 S Z A P C	
Ex	RPO 1 11/5 - - - - -	POP H 1 10 - - - - -	JPO a16 3 10 - - - - -	XTHL 1 18 - - - - -	CPO a16 3 17/11 - - - - -	PUSH H 1 11 - - - - -	ANI d8 2 7 S Z A P C		RPE 1 11/5 - - - - -	PCHL 1 5 - - - - -	JPE a16 3 10 - - - - -	XCHG 1 5 - - - - -	CPE a16 3 17/11 - - - - -	*CALL a16 3 17 - - - - -	XRI d8 2 7 S Z A P C	
Fx	RP 1 11/5 - - - - -	POP PSW 1 10 S Z A P C	JP a16 3 10 - - - - -		CP a16 3 17/11 - - - - -	PUSH PSW 1 11 - - - - -	ORI d8 2 7 S Z A P C		RM 1 11/5 - - - - -	SPHL 1 5 - - - - -	JM a16 3 10 - - - - -		CM a16 3 17/11 - - - - -	*CALL a16 3 17 - - - - -	CPI d8 2 7 S Z A P C	

Assembler

- 74 Mnemonics
- 2 Passes
- Implementiert in JS/TS mit pegjs
  - https://pegjs.org/
  - https://en.wikipedia.org/wiki/Parsing_expression_grammar
- Labels werden im ersten Pass gelesen
- 2 Pass übersetzt und ersetzt Labels mit Sprungadressen

Assembler - Beispiel

```
; Fib(12)

        MVI     A,1
        MVI     B,1
        MVI     D,BH
        LXI     H, MEM
LOOP:
        MOV     C,A
        MOV     M,A
        INX     H
        ADD     B
        MOV     B,C
        DCR     D
        JNZ     LOOP

        MOV     M,A
        HLT
MEM:
```

UI - Angular8

- Minimales UI
- geschrieben mit angular8
- 2 Teile
  - Assembler
  - Prozessor
- Code wird unmittelbar Compiliert und kann dann geladen werden
- Prozessorzustand kann beobachtet werden
- Taktfrequenzen
  - Rechnerabhängig
  - bzw. von 0.5Hz bis 200Hz(js engine limit *)

* https://developer.mozilla.org/en-US/docs/Web/API/WindowOrWorkerGlobalScope/setTimeout#Minimum_delay_and_timeout_nesting

Demo

http://i8080.sysvyz.org/