Emulacja procesora 8086 oparta na nowoczesnym standardzie języka C++

Patryk Kaniewski

December 1, 2021

Cel

Zbudowanie biblioteki do emulacji 8086 w C++20, a nastepnie zbudowanie na jej podstawie emulatora 8086 ktory mozna byłoby wykonywać proste programy DOS.

Używane technologie

- C++20
- GCC
- cmake

Demo

```
        mov ah, 0
:label
        add ah, 1   ;inc ah
        cmp ah, 12
        jne label
        mov ah, 0
        int 23h;
```

Obecne ograniczenia

- tylko normalne formy instrukcji
- ilość instrukcji
- ręczny assembler (formy instrukcji)
- brak grafiki

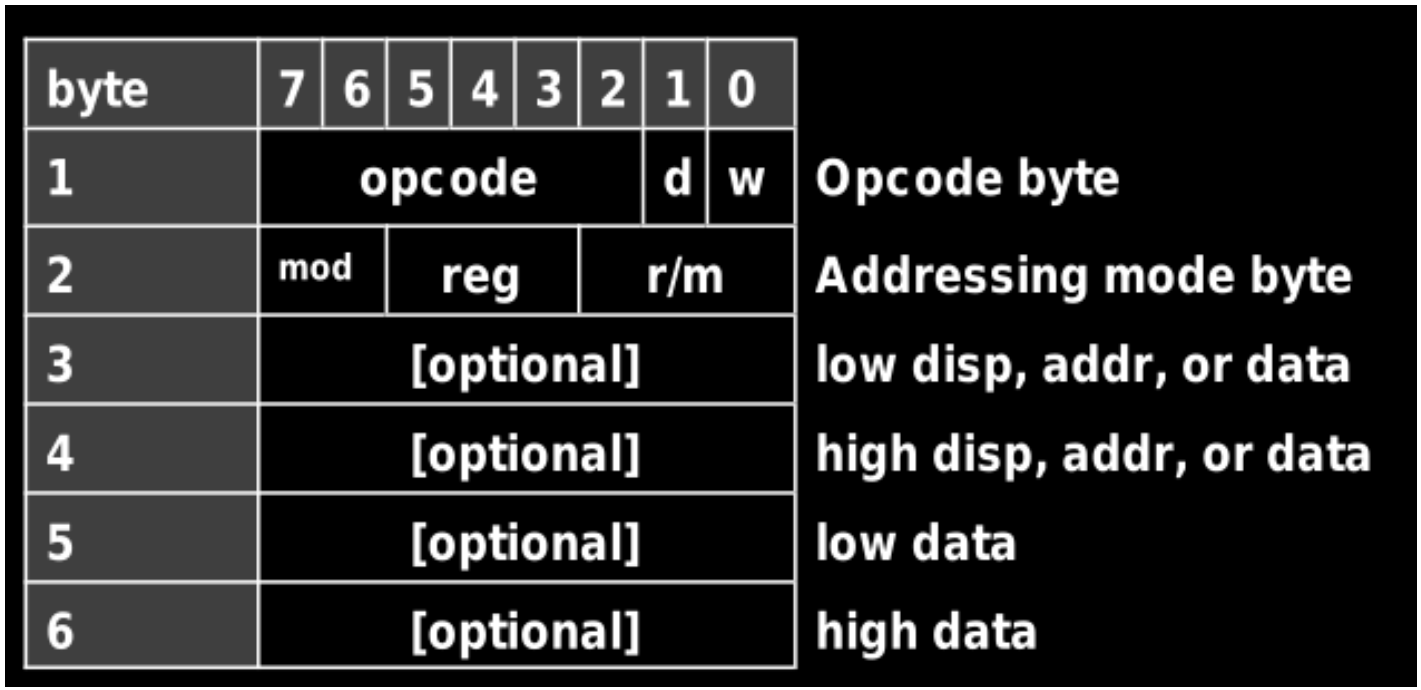
Napotkane problemy

- krótkie formy instrukcji

mnemonics	op xx xx xx xx xx	sw	len	flags
ADD AL,ib	04 i0	B	2	o---szap
ADD AX,iw	05 i0 i1	W	3	o---szap
ADD rb,rmb	02 mr d0 d1	B	2~4	o---szap
ADD rw,rmw	03 mr d0 d1	W	2~4	o---szap
ADD rmb,ib	80 /0 d0 d1 i0	NB	3~5	o---szap
ADD rmw,iw	81 /0 d0 d1 i0 i1	NW	4~6	o---szap
ADD rmw,ib	83 /0 d0 d1 i0	EW	3~5	o---szap
ADD rmb,rb	00 mr d0 d1	B	2~4	o---szap
ADD rmw,rw	01 mr d0 d1	W	2~4	o---szap

Napotkane problemu cd.

- wielotrybowe instrukcje (CMP/ADD)
  `ADD rm,imm8 vs CMP rm, imm8`



byte	7	6	5	4	3	2	1	0	
1	opcode						d	w	Opcode byte
2	mod		reg			r/m			Addressing mode byte
3	[optional]								low disp, addr, or data
4	[optional]								high disp, addr, or data
5	[optional]								low data
6	[optional]								high data

Miłe zaskoczenia

- adresowanie

```
void Interpreter::ADD_byte(Instruction& insn)
{
  printf("Interpreter::ADD_byte(%p)\n", &insn);
  int8_t src1 = insn.readFirstArgument<int8_t>();
  int8_t src2 = insn.readSecondArgument<int8_t>();

  int8_t dest = src1 + src2;

  flags_add<int8_t>(insn.flags(), src1 ,src2);
  printf("\tADD %d + %d = %d\n", src1,src2,dest);

  insn.writeSecondArgument<int8_t>(dest);
}
```

Obecne pytania

- rozszerzalność INT
- MMU visibility