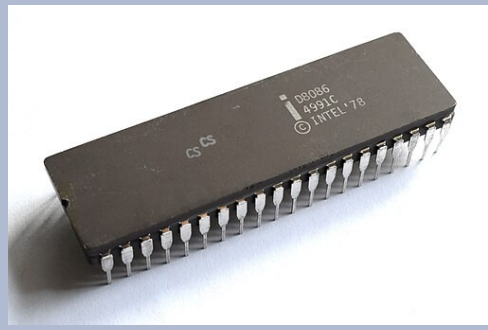
Pregled instrukcijske arhitekture x86

- Uvod
- Registri i instrukcije
- Prenošenje argumenata u potprograme
- Primjeri instrukcija

Siniša Šegvić
UniZg-FER

Instrukcijska arhitektura x86

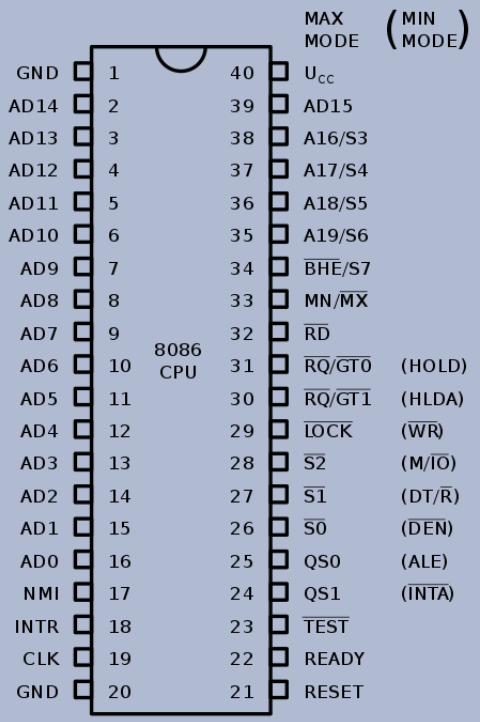


Nastala 1978. godine kad se pojavio Intel 8086
- proširenje arhitekture Intel 8080

Jeftina alternativa velikim mini-računalima:
- PDP, VAX, IBM S/360 itd.

Intel se pojavio u pravo vrijeme na pravom mjestu:
- memorijska tvrtka u zoru Mooreovog zakona

Jednostavno proširenje akumulatorske arhitekture:
- više instrukcija, više specijaliziranih registara (CISC)



Primjeri instrukcija

Prebaci registar `ax` u memoriju na adresu `bp-8`:

```
mov  WORD PTR -8[bp], ax
```

Prebaci adresu `bp`-40 u registar `dx`:

```
lea  dx, -40[bp]
```

CISC filozofija: specijalizirani registri, neortogonalne instrukcije:

- registar `bp` ne omogućava pristup višem i nižem bajtu
- registar `ax` ne može se pojaviti u uglatim zagradama
- instrukcija `sub dx,bp,#40` ne postoji

Arhitektura x86 poletjela je na krilima IBM-ovog PC-ja, međutim ranih 1990-tih pojavili su se brojni izazivači:

- Alpha, MIPS, SUN Sparc, ARM, ...

CISC vs RISC

Ranih 1990-tih pojavile su se RISC arhitekture (Alpha, ...):

- (+) učinkovitije za budžete od 100k tranzistora
- (-) nisu mogle izvršavati postojeće programe...

Unatoč lošim izgledima, CISC je odgovorio 1995. godine

- ideja: dinamički prevoditi CISC instrukcije (x86) u interne RISC instrukcije (CISC ISA, RISC organizacija)
- PentiumPro, Cyrix 6x86 i AMD K5

Razlozi uspjeha:

- veliki moment uslijed rasprostranjenosti, Mooreov zakon
- prednost “čistih” arhitektura (MIPS, Sparc, Alpha) poništena kompetitivnom tehnologijom

Registri „opće namjene”:

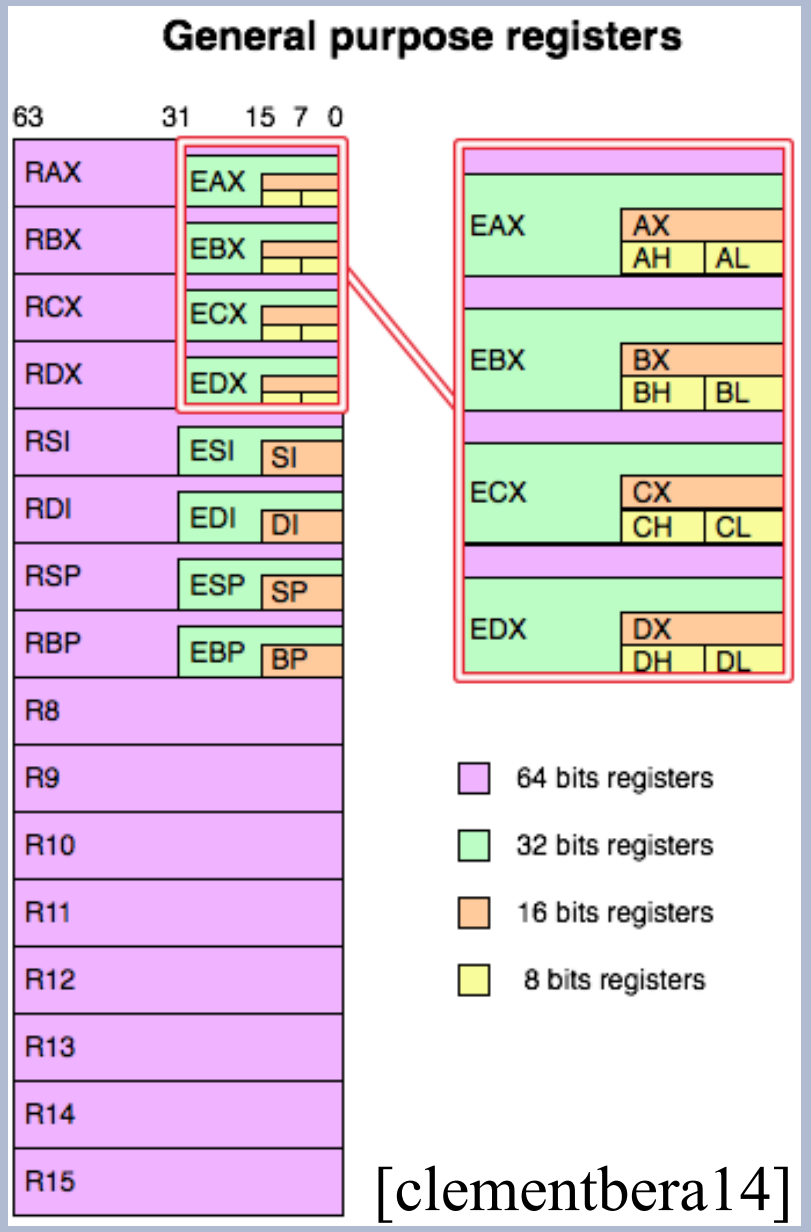
- ah,al (8b), ax (16b)
- eax (32b), rax (64b)
- donja polovica od ax je al
- jednako za eax i ax te rax i eax.
- i8080 i8086 i80386 Opteron (golden handcuffs)

Registri segmenata

- CS, SS, DS, ES, FS, GS
- nemaju praktičnu važnost (atavizam)

Upravljački registri:

- programsko brojilo: IP, EIP, RIP
- registar stanja: FLAGS, EFLAGS, ...



[clementbera14]

Oblici instrukcija: dvoadresni akumulacijski oblik

- osim instrukcije `lea` i vektorskih instrukcija koje su tro- ili više-adresne

Source/destination operand type	Second source operand
Register	Register
Register	Immediate
Register	Memory
Memory	Register
Memory	Immediate

[patterson13]

Tipične instrukcije

Instruction	Function
je name	if equal(condition code) {EIP=name}; EIP-128 <= name < EIP+128
jmp name	EIP=name
call name	SP=SP-4; M[SP]=EIP+5; EIP=name;
push ESI	SP=SP-4; M[SP]=ESI
pop EDI	EDI=M[SP]; SP=SP+4
add EAX,#6765	EAX= EAX+6765
test EDX,#42	Set condition code (flags) with EDX and 42
movsl	M[EDI]=M[ESI]; EDI=EDI+4; ESI=ESI+4

[patterson13]

Standard za povezivanje izvršnih programa (ABI)

Standard za povezivanje izvršnih programa (eng. application binary interface) najvažniji je standard za kojeg nikad niste čuli:

- instrukcijsku arhitekturu
- veličine (sizeof), rasporede (big/little endian) i poravnanja (mod 2, 4, 8 itd) osnovnih tipova podataka
- **prijenos argumenata u potprograme**
- protokol poziva usluga operacijskog sustava
- formate izvršnih datoteka i biblioteka

Prijenos argumenata u potprograme

System V AMD64: ABI za UNIX-like sustave

- Solaris, Linux, FreeBSD, macOS
- većina argumenata prenosi se preko registara (tablica dolje)
- pozvana funkcija (eng. callee) može po volji mijenjati registre osim rbx, rsp, rbp i r12–r15

Tip	Registri
Cjelobrojni/pokazivači 1-6	rdi, rsi, rdx, rcx, r8, r9
Decimalni 1-8	xmm0 - xmm7
Višak argumenata	stog

Sličan standard postoji za Windowse (Microsoft x64)

- detalji se razlikuju

Implementiranje potprograma

Inicijaliziraj pokazivač okvira (`rbp`):

```
push rbp
mov rbp, rsp
```

Zauzmi prostor na stogu (poravnanje 16b!):

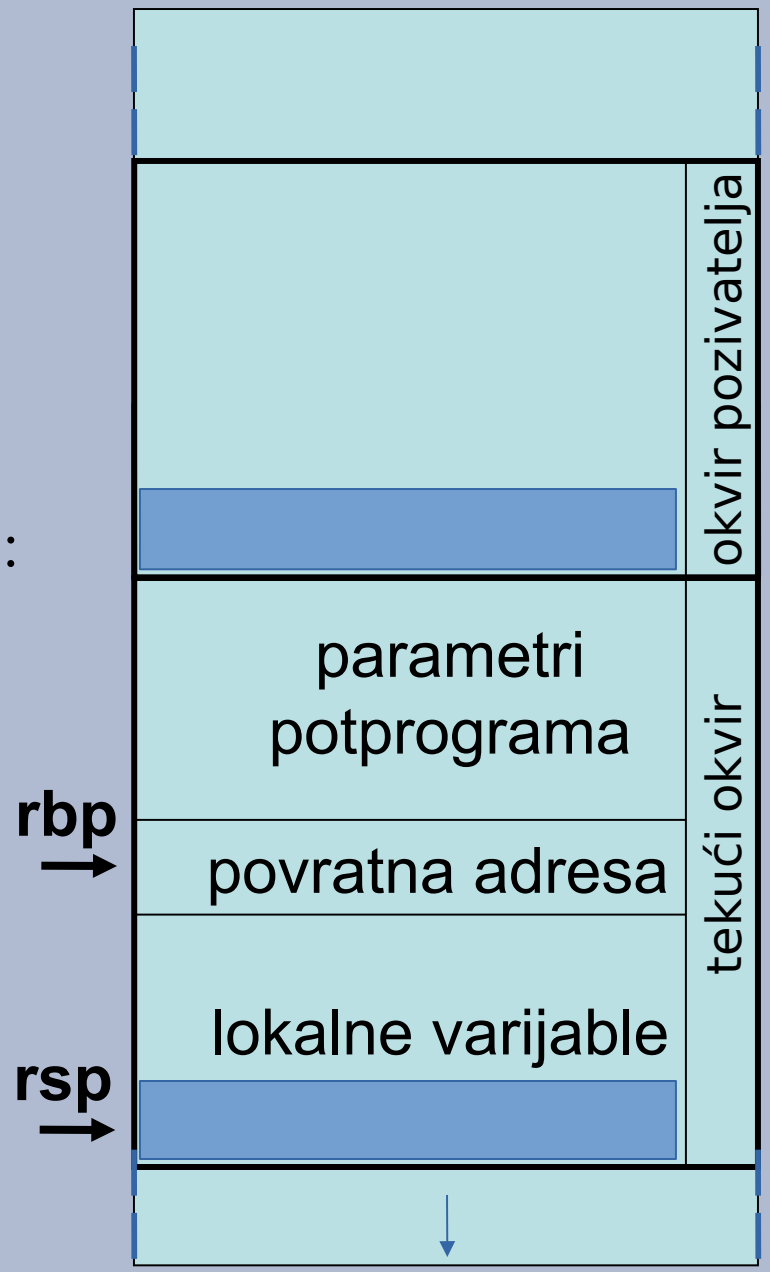
```
sub rsp, 16
```

Prebaci argument (`rdi`) u lokalnu var.:

```
mov QWORD PTR -8[rbp], rdi
```

Postavi povratnu vrijednost i vrati se:

```
mov eax, 42
leave # rsp←rbp, pop rbp
ret
```



Još primjera

Pribroji statičku varijablu (relativno adresiranje!):

```
add  eax, DWORD PTR [rip+<32b-offset>]
```

Dohvati adresu relativno adresiranog podatka:

```
lea  rdx, [rip+<32b-offset>]
```

Poziv potprograma:

```
call <address>
call rdx
```

Upiši u `rax` podatak na kojeg `rax` pokazuje (p=*p):

```
mov  rax, QWORD PTR [rax]
```

Još primjera (2)

Postavljanje registra na nulu:

```
xor  rax, rax # 3B (2B) na x86-64(32)
mov  rax, 0   # 7B (5B) na x86-64(32)
```

Skalirano indeksiranje, uvećavanje, troadresno zbrajanje:

```
add  rax, QWORD PTR [rdi+24+rsi*8]
inc  QWORD PTR [rsp+70h]
lea  rax, [rbx+rcx]
```

Optimizirano tijelo funkcije iabs (fali samo `ret`):

```
mov rax, rdi
neg rax
cmovl rax, rdi
```