nedeljainformatike v7.0

Dekompilacija: kako razumeti asembler

Momčilo Topalovic

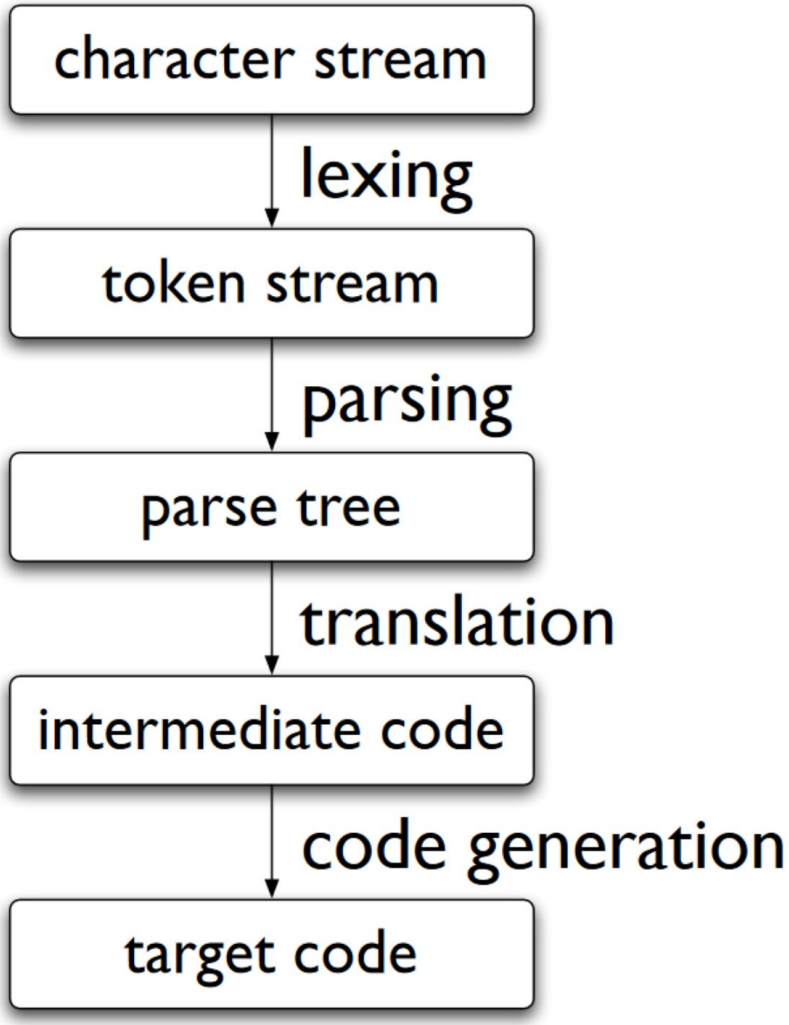
Matematička gimnazija

14. 04. 2022.

Agenda

1. Kompilacija
2. Dekompilacija
3. Realnost

Faze kompilacije



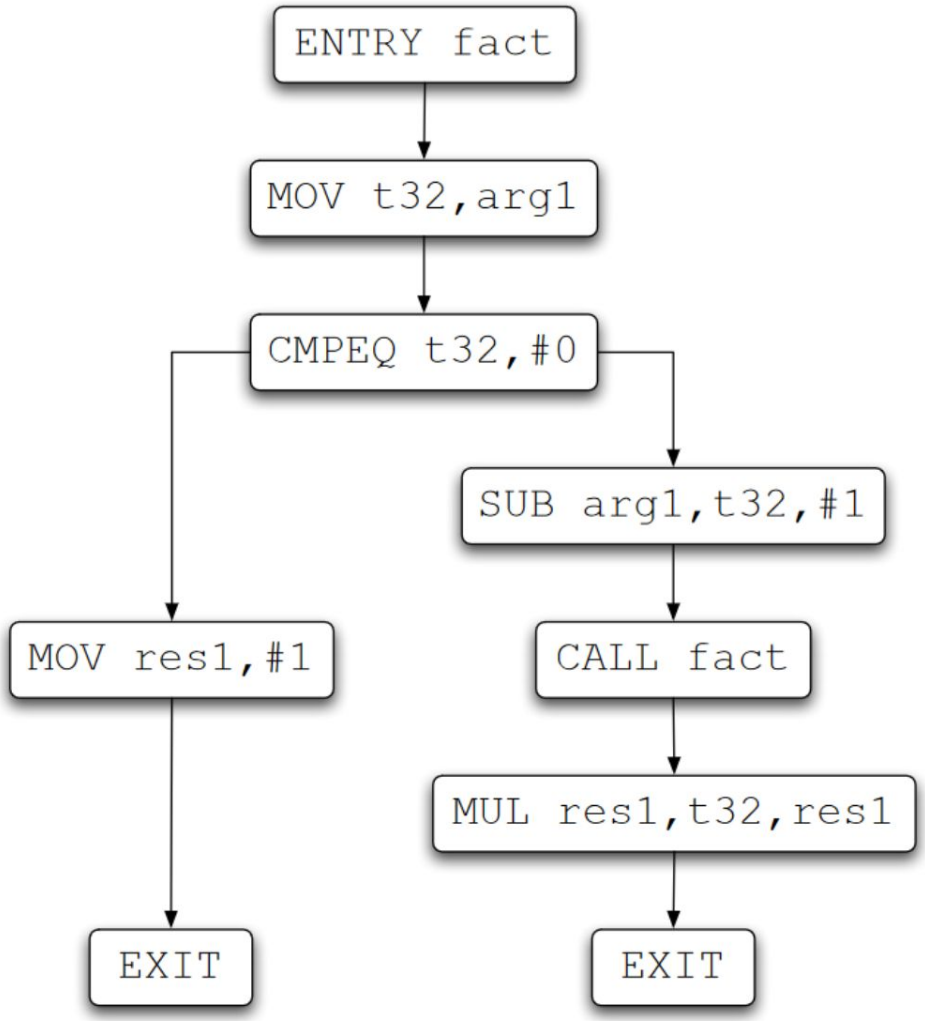
Intermediate Code

```
int  fact (int n) {
    if (n == 0)
        return 1;
    else
        return n * fact(n - 1);
}
```

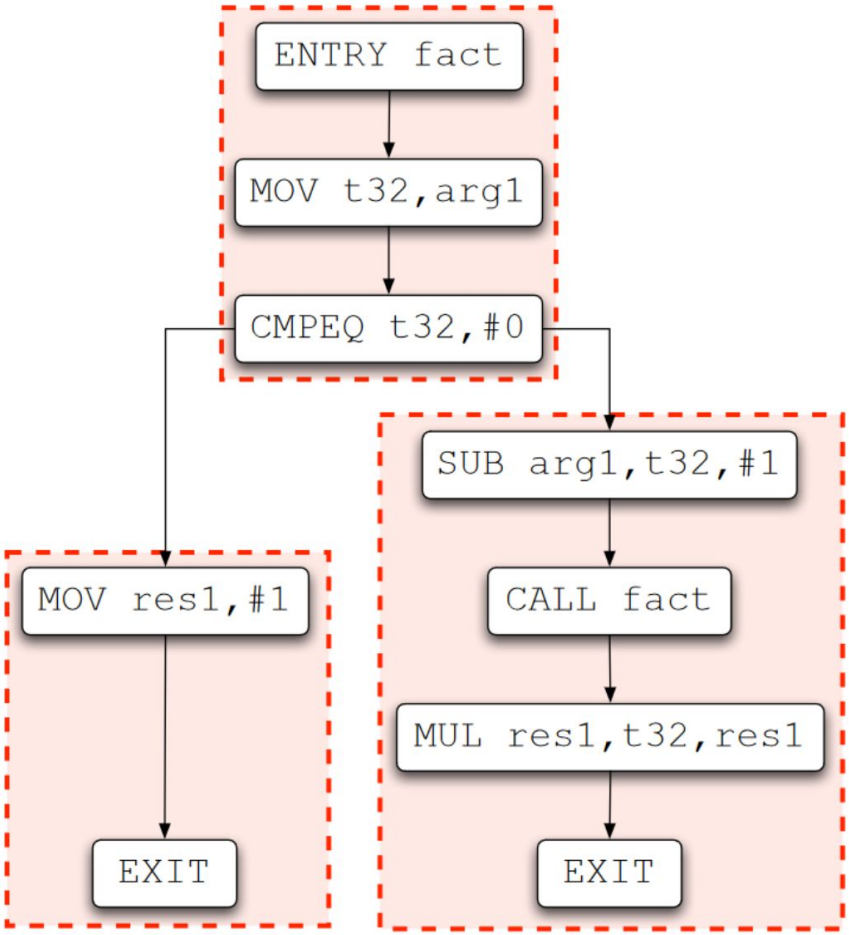
Intermediate Code, cont'd

```
        ENTRY   fact
        MOV   t32, arg1
        CMPEQ   arg1, #0, lab1
        SUB   arg1, t32, #1
        CALL   fact
        MUL   res1, t32, res1
        EXIT
lab1:   MOV   res1, #1
        EXIT
```

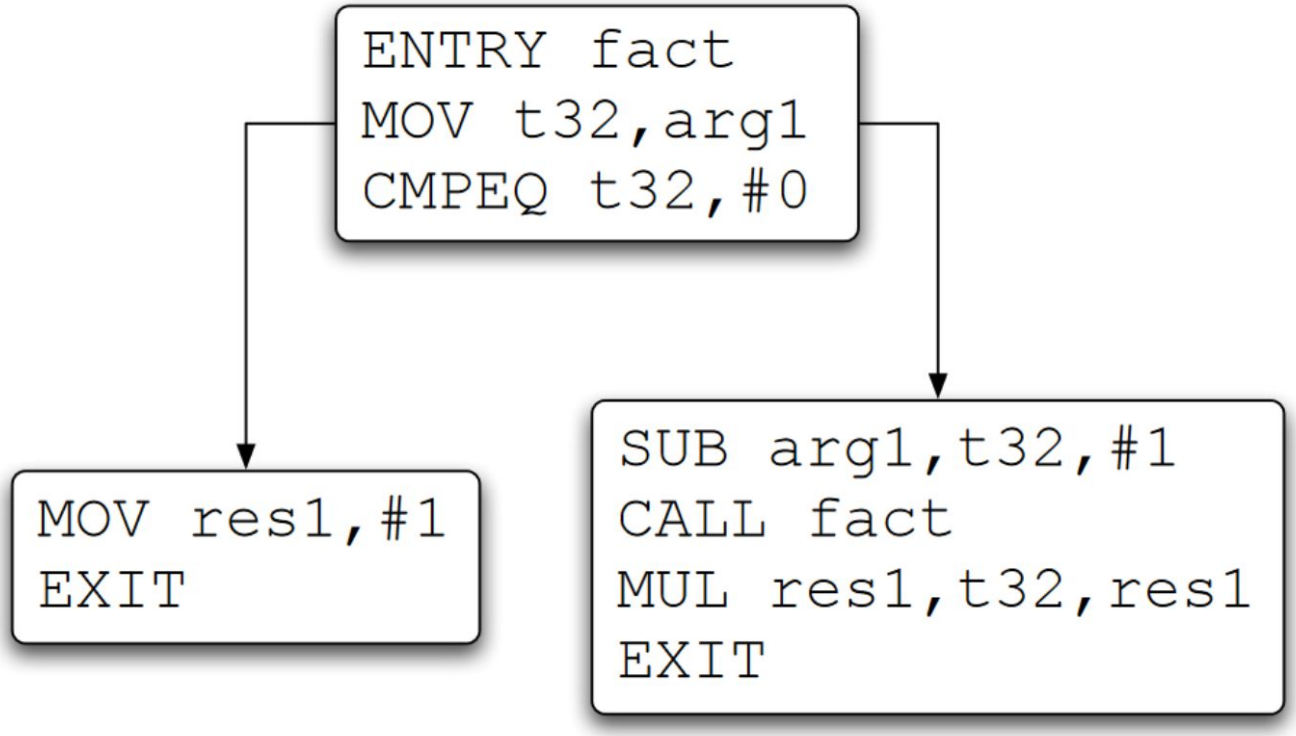
Flow Graph


ENTRY fact
MOV t32,arg1
CMPEQ t32,#0
SUB arg1,t32,#1
MOV res1,#1
CALL fact
MUL res1,t32,res1
EXIT
EXIT

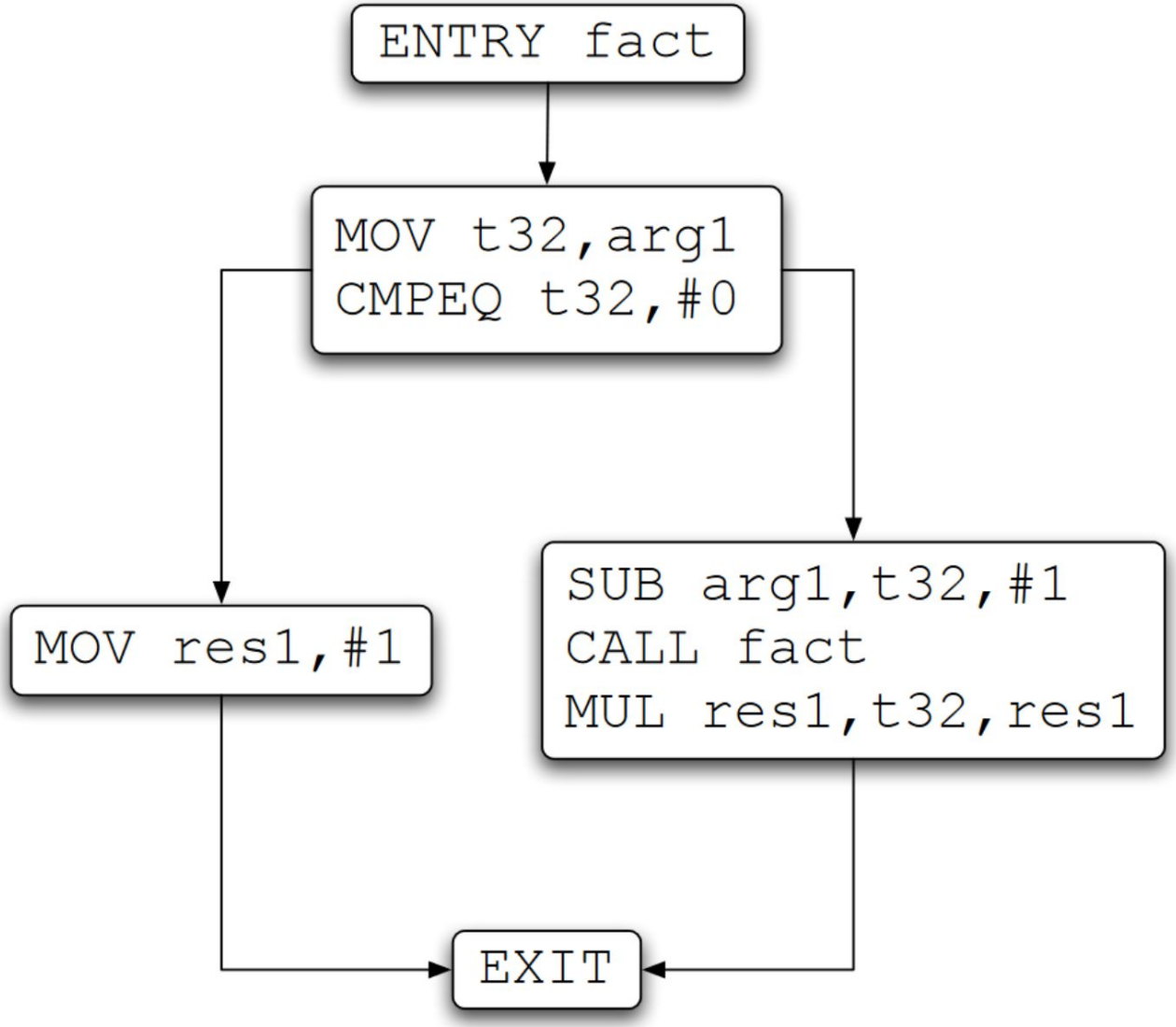
Flow Graph, cont'd



Flow Graph, cont'd



Flow Graph, cont'd



Šta možemo da obrišemo?

```
const bool debug = false;

void foo(int n) {
    n = n;

    if (false)
        printf(“ 1 \n”);

    if (debug)
        printf(“ 2 \n”);

    if (prime(n) && prime(n + 1) && n != 2)
        printf(“ 3 \n”);
}
```

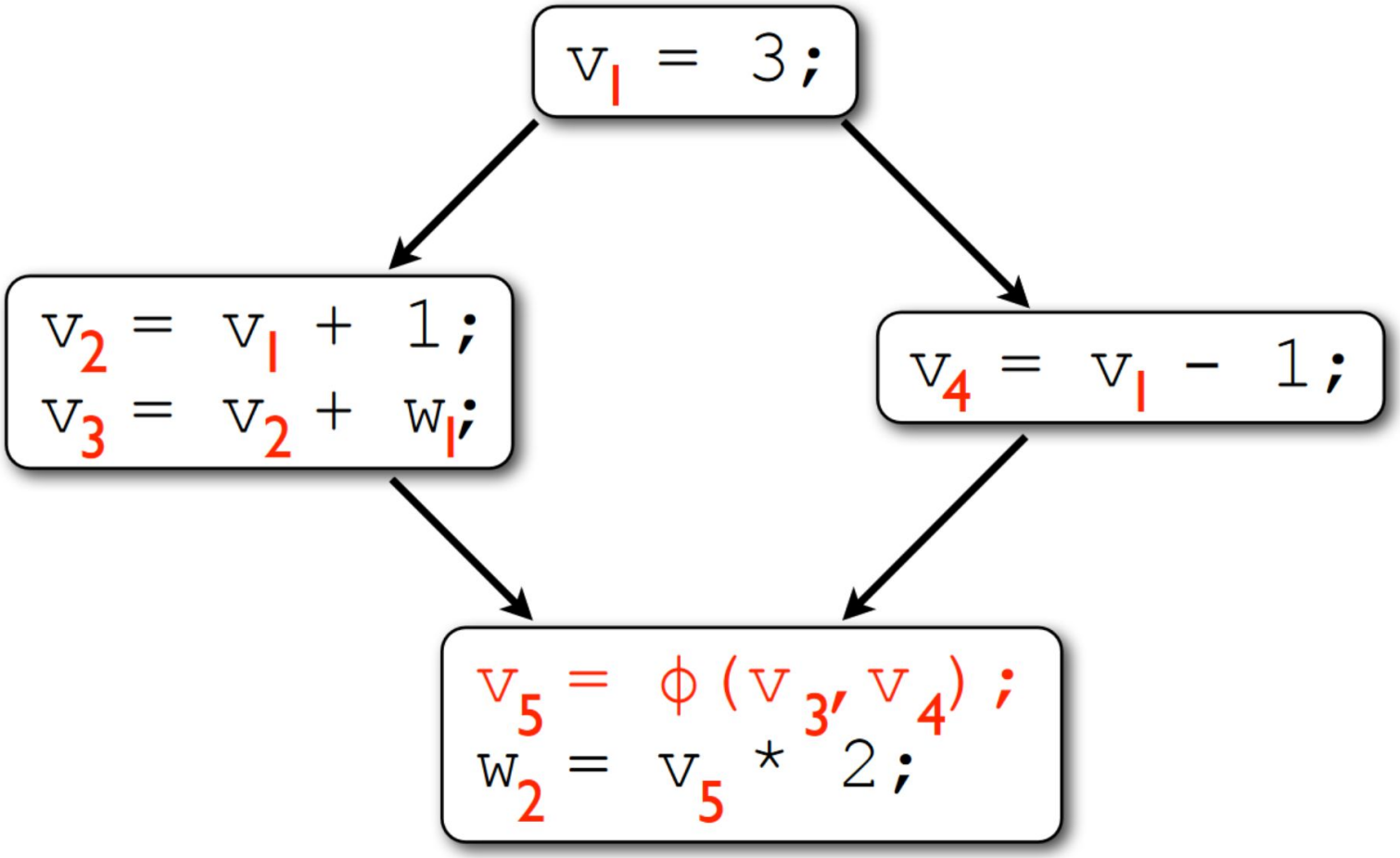
SSA forma

$$x = 0$$
$$v = 3$$
$$v = v + 1$$
$$v = v + x$$
$$x = v + 2$$

SSA forma, cont'd

$$x\_1 = 0$$
$$v\_1 = 3$$
$$v\_2 = v\_1 + 1$$
$$v\_3 = v\_2 + x\_1$$
$$x\_2 = v\_3 + 2$$

SSA forma



Još optimizacija

- Instruction scheduling
- Variable allocation

Agenda

1. Kompilacija
2. Dekompilacija
3. Realnost

Nezanimljiv deo

- Učitavanje binarne datoteke
- Pronalaženje funkcija i globalnih promenljivih
- Pronalaženje biblioteka
- Prevođenje u među-jezik

Flow Graph Reconstruction (demo)

Flow Graph Reconstruction

Ključne ideje:

- Dodavanje novih čvorova
- Grananje - dominatori i porodica
- Petlje - jako povezane komponente

Rekonstrukcija tipova (demo)

Rekonstrukcija tipova

Ključne ideje:

- Svaka naredba nam daje neka ograničenja
- Tražimo dodelu koja zadovoljava sva ograničenja

Trenutno

```
int fact(int n0) {
        int v1 = 1;
        bool v2 = ((n0 & n0) != 0);
        if (v2) {
                int v3 = n0;
                n1 = n0 - 1;
                int v4 = fact(n1);
                v1 = v1 * v4;
                return v1;
        }
        else {
                return v1;
        }
}
```

Agenda

1. Kompilacija
2. Dekompilacija
3. Tužna realnost

Divne stvari o kojima nismo pričali

- Pozivanje funkcija
- Stack
- Pointers
- Nasleđivanje
- Dinamični skokovi
- Dinamične funkcije

Divan primer arhitekture

- x86-64 DIV (unsigned divide)

Operand Size	Dividend	Divisor	Quotient	Remainder	Maximum Quotient
Word/byte	AX	r/m8	AL	AH	255
Doubleword/word	DX:AX	r/m16	AX	DX	65,535
Quadword/doubleword	EDX:EAX	r/m32	EAX	EDX	$2^{32} - 1$
Doublequadword/quadword	RDX:RAX	r/m64	RAX	RDX	$2^{64} - 1$

Divan primer optimizacije

void misterija (int *a, int *b, int n);

```
misterija(int*, int*, int):
        test    edx, edx
        jle     .L1
        lea     rcx, [rsi+4]
        mov     rax, rdi
        sub     rax, rcx
        cmp     rax, 8
        jbe     .L3
        lea     eax, [rdx-1]
        cmp     eax, 3
        jbe     .L3
        mov     ecx, edx
        xor     eax, eax
        shr     ecx, 2
        sal     rcx, 4
.L4:
        movdqu  xmm0, XMMWORD PTR [rsi+rax]
        movups  XMMWORD PTR [rdi+rax], xmm0
        add     rax, 16
        cmp     rax, rcx
        jne     .L4
        mov     eax, edx
        and     eax, -4
        test    dl, 3
        je      .L1
        mov     ecx, eax
        mov     r8d, DWORD PTR [rsi+rcx*4]
        mov     DWORD PTR [rdi+rcx*4], r8d
        lea     ecx, [rax+1]
        cmp     edx, ecx
        jle     .L1
        movsx   rcx, ecx
        add     eax, 2
        mov     r8d, DWORD PTR [rsi+rcx*4]
        mov     DWORD PTR [rdi+rcx*4], r8d
        cmp     edx, eax
        jle     .L1
        cdqe
        mov     edx, DWORD PTR [rsi+rax*4]
        mov     DWORD PTR [rdi+rax*4], edx
        ret
.L3:
        mov     edx, edx
        xor     eax, eax
.L7:
        mov     ecx, DWORD PTR [rsi+rax*4]
        mov     DWORD PTR [rdi+rax*4], ecx
        add     rax, 1
        cmp     rax, rdx
        jne     .L7
.L1:
        ret
```

Divan primer optimizacije

```
void misterija (int *a, int *b, int n) {
    for (int i = 0; i < n; i++)
        a[i] = b[i];
}
```

```
misterija(int*, int*, int):
        test    edx, edx
        jle     .L1
        lea     rcx, [rsi+4]
        mov     rax, rdi
        sub     rax, rcx
        cmp     rax, 8
        jbe     .L3
        lea     eax, [rdx-1]
        cmp     eax, 3
        jbe     .L3
        mov     ecx, edx
        xor     eax, eax
        shr     ecx, 2
        sal     rcx, 4
.L4:
        movdqu  xmm0, XMMWORD PTR [rsi+rax]
        movups  XMMWORD PTR [rdi+rax], xmm0
        add     rax, 16
        cmp     rax, rcx
        jne     .L4
        mov     eax, edx
        and     eax, -4
        test    dl, 3
        je      .L1
        mov     ecx, eax
        mov     r8d, DWORD PTR [rsi+rcx*4]
        mov     DWORD PTR [rdi+rcx*4], r8d
        lea     ecx, [rax+1]
        cmp     edx, ecx
        jle     .L1
        movsx   rcx, ecx
        add     eax, 2
        mov     r8d, DWORD PTR [rsi+rcx*4]
        mov     DWORD PTR [rdi+rcx*4], r8d
        cmp     edx, eax
        jle     .L1
        cdqe
        mov     edx, DWORD PTR [rsi+rax*4]
        mov     DWORD PTR [rdi+rax*4], edx
        ret
.L3:
        mov     edx, edx
        xor     eax, eax
.L7:
        mov     ecx, DWORD PTR [rsi+rax*4]
        mov     DWORD PTR [rdi+rax*4], ecx
        add     rax, 1
        cmp     rax, rdx
        jne     .L7
.L1:
        ret
```

Hvala na pažnji!

Pitanja?