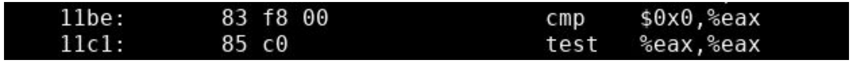
АКОС

Семинар 3
15.11.2021

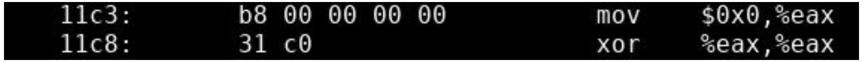
Длина инструкций

1. cmp vs test:



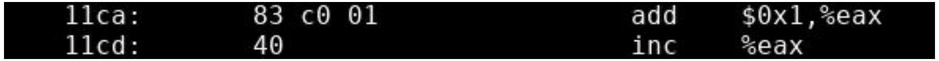
```
11be:      83 f8 00                  cmp    $0x0,%eax
11c1:      85 c0                     test   %eax,%eax
```

2. mov vs xor:



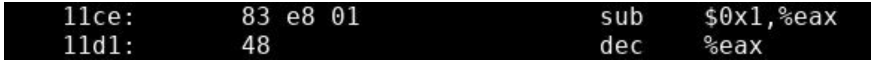
```
11c3:      b8 00 00 00 00            mov    $0x0,%eax
11c8:      31 c0                     xor    %eax,%eax
```

3. add vs inc:



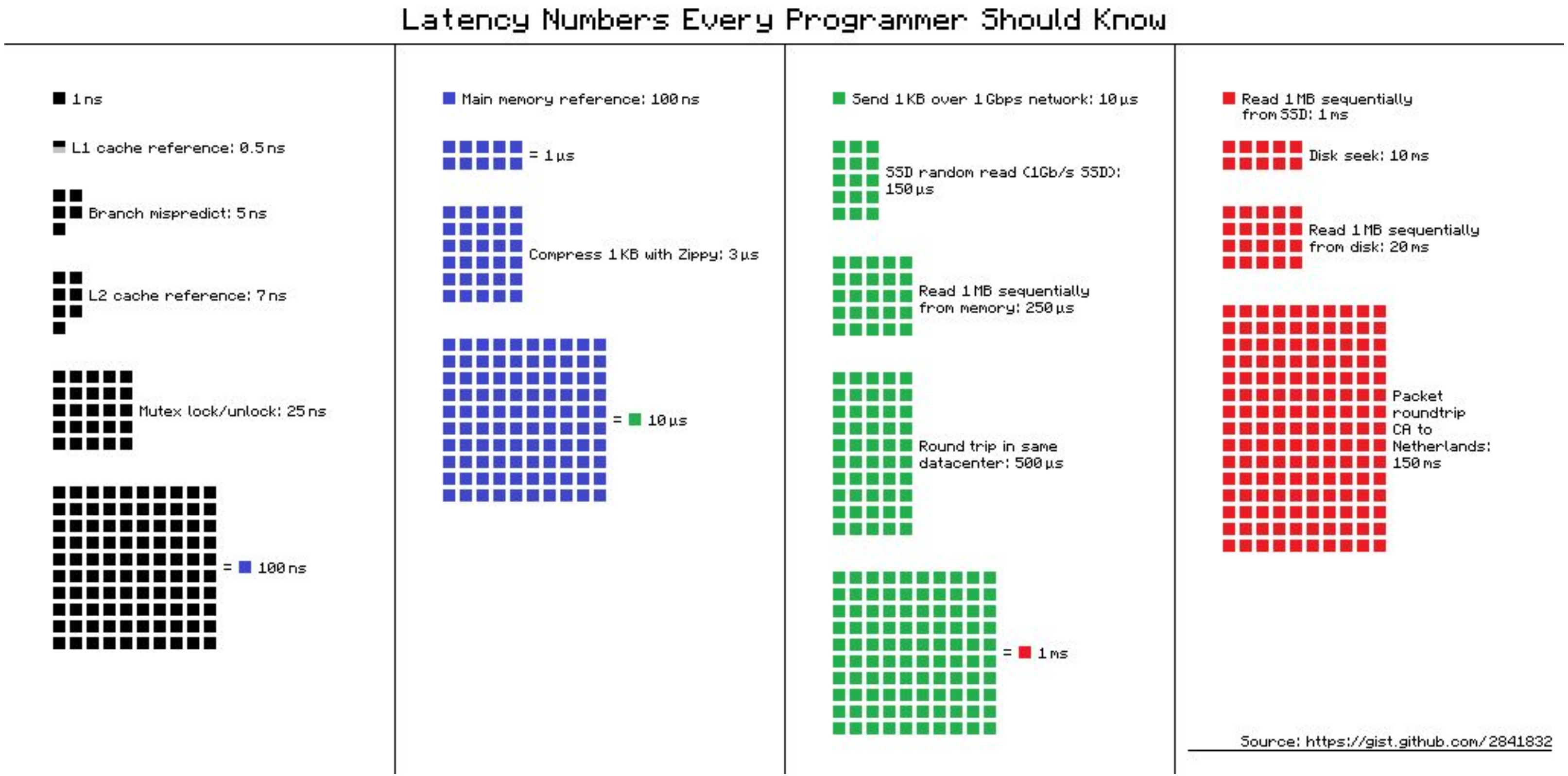
```
11ca:      83 c0 01                  add    $0x1,%eax
11cd:      40                        inc    %eax
```

4. sub vs dec:



```
11ce:      83 e8 01                  sub    $0x1,%eax
11d1:      48                        dec    %eax
```

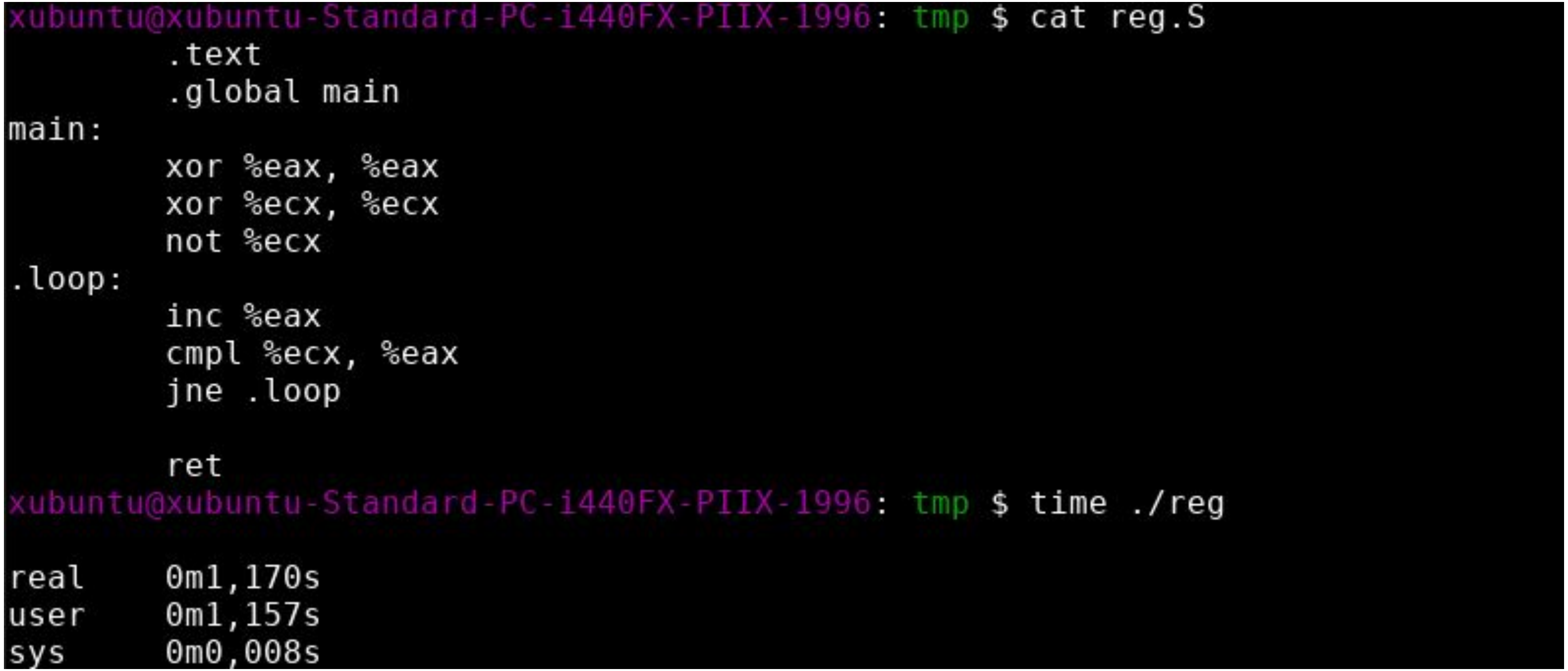
Numbers Everybody Should Know



Latency Numbers Every Programmer Should Know

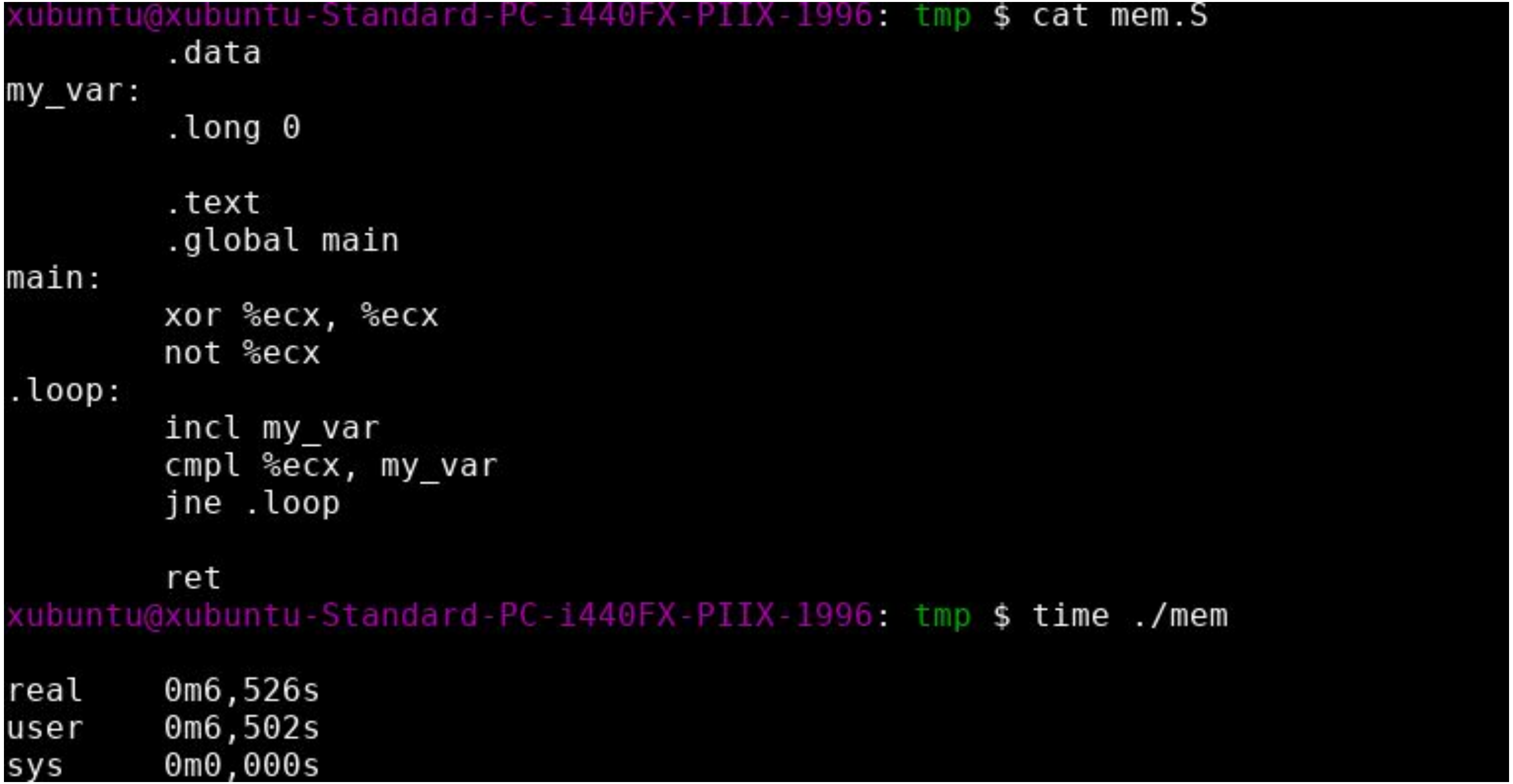
Source: https://gist.github.com/2841832

Очень грубый бенчмарк



```
xubuntu@xubuntu-Standard-PC-i440FX-PIIX-1996: tmp $ cat reg.S
        .text
        .global main
main:
        xor %eax, %eax
        xor %ecx, %ecx
        not %ecx
.loop:
        inc %eax
        cmpl %ecx, %eax
        jne .loop

        ret
xubuntu@xubuntu-Standard-PC-i440FX-PIIX-1996: tmp $ time ./reg

real    0m1,170s
user    0m1,157s
sys     0m0,008s
```

Очень грубый бенчмарк

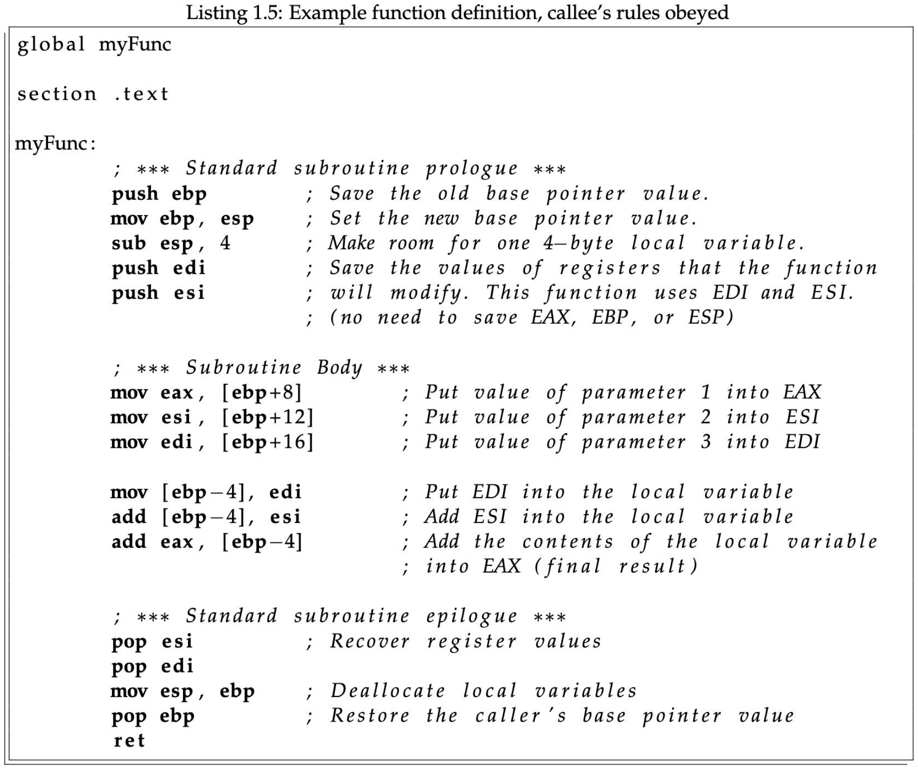


```
xubuntu@xubuntu-Standard-PC-i440FX-PIIX-1996: tmp $ cat mem.S
        .data
my_var:
        .long 0

        .text
        .global main
main:
        xor %ecx, %ecx
        not %ecx
.loop:
        incl my_var
        cmpl %ecx, my_var
        jne .loop

        ret
xubuntu@xubuntu-Standard-PC-i440FX-PIIX-1996: tmp $ time ./mem

real    0m6,526s
user    0m6,502s
sys     0m0,000s
```

Prologue / Epilogue

```
push %ebp // Сохраняем указатель на фрейм вызывающей функции

mov %esp, %ebp // Фрейм нашей функции начинается с текущего места в стеке, т.е. с вершины

…

mov %ebp, %esp // Согласно cdecl, функция должна вернуть указатель на стек к исходному состоянию

pop %ebp // Восстанавливаем ebp
```

Stack frame

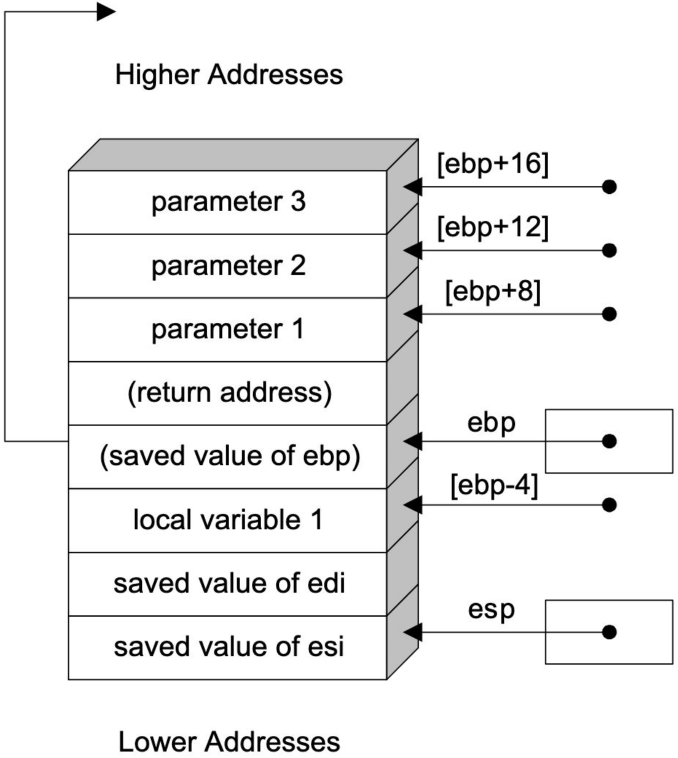


Listing 1.5: Example function definition, callee's rules obeyed

```
global myFunc

section .text

myFunc:
        ; *** Standard subroutine prologue ***
        push ebp          ; Save the old base pointer value.
        mov ebp, esp      ; Set the new base pointer value.
        sub esp, 4        ; Make room for one 4-byte local variable.
        push edi          ; Save the values of registers that the function
        push esi          ; will modify. This function uses EDI and ESI.
                          ; (no need to save EAX, EBP, or ESP)

        ; *** Subroutine Body ***
        mov eax, [ebp+8]          ; Put value of parameter 1 into EAX
        mov esi, [ebp+12]         ; Put value of parameter 2 into ESI
        mov edi, [ebp+16]         ; Put value of parameter 3 into EDI

        mov [ebp-4], edi          ; Put EDI into the local variable
        add [ebp-4], esi          ; Add ESI into the local variable
        add eax, [ebp-4]          ; Add the contents of the local variable
                                  ; into EAX (final result)

        ; *** Standard subroutine epilogue ***
        pop esi           ; Recover register values
        pop edi
        mov esp, ebp      ; Deallocate local variables
        pop ebp           ; Restore the caller's base pointer value
        ret
```

https://aaronbloomfield.github.io/pdr/book/x86-32bit-ccc-chapter.pdf

Stack frame



https://aaronbloomfield.github.io/pdr/book/x86-32bit-ccc-chapter.pdf