Gabriele **Proietti Mattia**

Advanced Operating Systems and Virtualization

[Lab 03] ASM in C

Department of Computer, Control and Management Engineering "A. Ruberti", Sapienza University of Rome

gpm.name · proiettimattia@diag.uniroma1.it

A.Y. 2020/2021 · v1

Inline Assembly

C programs can contain assembly code. The structure of the command is the following:

ASM Directive

Volatile Directive
Tells the compiler to leave the code here

ASM Instructions

```
__asm__ __volatile__ ( "assembly code"
                     : output operands                  /* optional */
                     : input operands                   /* optional */
                     : list of clobbered registers      /* optional */
);
```

- **Output Operands**: comma separated list of inputs, e.g. `"=r" (old), "+rm" (*Base)`
- **Input Operands**: comma separated list of output , e.g. `"r" (Offset)`
- **Clobbers**: comma separated list of registers or other elements that have been changed by the execution of the instruction(s) (e.g. GCC won't use these registers to store any other value).

Legend

Operands

- `"m"`: a memory operand
- `"o"`: a memory operand which is "offsettable" (to deal with instructions' size)
- `"r"`: a general-purpose register
- `"g"`: Register, memory or immediate, except for non-general purpose registers
- `"i"`: an immediate operand

Registers

- `"0", "1", … "9"`: a previously referenced register
- `"q"`: any "byte-addressable" register
- `"Q"` any “high” 8-bit addressable sub-register
- `"+"`: the register is both read and written
- `"="`: the register is written
- `"a", "b", "c", "d", "S", "D"`: registers A, B, C, D, SI, and DI
- `"A"`: registers A and D (for instructions using AX:DX as output)

Legend

r	Register(s)
a	%rax, %eax, %ax, %al
b	%rbx, %ebx, %bx, %bl
c	%rcx, %ecx, %cx, %cl
d	%rdx, %edx, %dx, %dl
S	%rsi, %esi, %si
D	%rdi, %edi, %di

Examples

https://github.com/gabrielepmattia/aosv-code-examples/tree/main/00-asm-in-c

CPUID

The CPUID assembly instruction allows to retrieve information about the available hardware.

```
__asm__("cpuid"
        : "=c"(*c), "=d"(*d)
        : "a"(code));
```

https://www.felixcloutier.com/x86/cpuid

wrmsr/rdmsr

A **model-specific register** (**MSR**) is any of various control registers in the x86 instruction set used for debugging, program execution tracing, computer performance monitoring, and toggling certain CPU features.

```
static inline void wrmsr(uint32_t msr_id, uint64_t msr_value) {
    __asm__ __volatile__ ( "wrmsr" : : "c" (msr_id), "A" (msr_value) );
}

static inline uint64_t rdmsr(uint32_t msr_id) {
    uint64_t msr_value;
    __asm__ __volatile__ ( "rdmsr" : "=A" (msr_value) : "c" (msr_id) );
    return msr_value;
}
```

Advanced Operating Systems and Virtualization

[Lab 03] Kernel Modules

LECTURER

Gabriele **Proietti Mattia**

gpm.name · proiettimattia@diag.uniroma1.it

DIAG