

## Inline Assembler Template

```
asm ( <instructions>
    : {output operands}
    : {input operands}
    : {clobbered registers}
    );
```

Empty groups still require a colon, except the group of clobbered registers

```
asm ( <instructions>
    : {output operands}
    :
    );
```

The `volatile` keyword prevents `asm` constructs from being moved, deleted or rescheduled.

```
asm volatile ( .... );
```

## Constraints

Describe valid operands and are denoted by letters:

- 'f': A floating-point register
- 'i': Immediate (constant) operands
- 'm': Memory operand
- 'r': A general-purpose register

## Inline Routines as Macros

```
#include <stdio.h>

#define readb(address, v) ({ \
    __asm__ volatile ("movb (%edi), %%al\n" \
        : "=a" (v) \
        : "D" (address)); \
})

int main(int argc, char *argv[]) {
    char x;
    char y;

    x = 'a';
    y = 'b';
    readb(&x, y);
    printf("%c\n", y);
    return 0;
}
```

The statement `readb(&x, y)` expands to

```
        leal    -5(%ebp), %edi
#APP
        movb    (%edi), %al

#NO_APP
        movb    %al, -6(%ebp)
\end{slide}
```

Alternatively, the macro can be designed to act like a function:

```
#define readb(address) ({ \
    unsigned char _v; \
    __asm__ volatile ("movb (%%edi), %%al\n" \
        : "=a" (_v) \
        : "D" (address)); \
    _v; \
}) */
```

## Inline Routines as C Functions

```
#include <stdio.h>

unsigned long long rdtsc() {
    unsigned long long tsc;
    asm volatile ("rdtsc":"=A" (tsc):);
    return tsc;
}

int main(int argc, char *argv[]) {
    printf("clock cycle count = %llu\n", rdtsc());
    return 0;
}
```

The rdtsc function:

```
rdtsc:
    pushl    %ebp
    movl    %esp, %ebp
    subl    $8, %esp
#APP
    rdtsc
#NO_APP
    movl    %eax, -8(%ebp)
    movl    %edx, -4(%ebp)
    movl    -8(%ebp), %eax
    movl    -4(%ebp), %edx
    leave
    ret
```

## Keywords

- `asm` not always recognized (`-traditional` and `-ansi` options)
- Use alternative keywords: `__asm__`
- Not always defined in other compilers — use macro definitions

```
#ifndef __GNUC__  
#define __asm__ asm  
#endif
```