X86 workshop

AsmMasters

Winter 2024

Using scanf & printf

Register	Usage	callee saved
`%rax`	temporary register; with variable arguments passes information about the number of vector registers used; 1st return register	No
`%rbx`	callee-saved register	Yes
`%rcx`	used to pass 4th integer argument to functions	No
`%rdx`	used to pass 3rd argument to functions; 2nd return register	No
`%rsp`	stack pointer	Yes
`%rbp`	callee-saved register; optionally used as frame pointer	Yes
`%rsi`	used to pass 2nd argument to functions	No
`%rdi`	used to pass 1st argument to functions	No
`%r8`	used to pass 5th argument to functions	No
`%r9`	used to pass 6th argument to functions	No
`%r10`	temporary register, used for passing a function's static chain pointer	No
`%r11`	temporary register	No
`%r12–%r14`	callee-saved registers	Yes
`%r15`	callee-saved register; optionally used as GOT base pointer	Yes
`%r16–%r31`	temporary registers	No
`%xmm0–%xmm1`	used to pass and return floating point arguments	No
`%xmm2–%xmm7`	used to pass floating point arguments	No
`%xmm8–%xmm15`	temporary registers	No
`%xmm16–%xmm31`	temporary registers	No
`%tmm0–%tmm7`	temporary registers	No
`%mm0–%mm7`	temporary registers	No
`%k0–%k7`	temporary registers	No
`%st0,%st1`	temporary registers, used to return `long double` arguments	No
`%st2–%st7`	temporary registers	No
`%fs`	thread pointer	Yes
`mxcsr`	SSE2 control and status word	partial
`x87 SW`	x87 status word	No
`x87 CW`	x87 control word	Yes
`tilecfg`	Tile control register	No

Classification The size of each argument gets rounded up to eightbytes.[14]
The basic types are assigned their natural classes:

- Arguments of types (signed and unsigned) `_Bool`, `char`, `short`, `int`, `long`, `long long`, and pointers are in the INTEGER class.
- Arguments of types `_Float16`, `float`, `double`, `_Decimal32`, `_Decimal64` and `__m64` are in class SSE.
- Arguments of types `__float128`, `_Decimal128` and `__m128` are split into two halves. The least significant ones belong to class SSE, the most significant one to class SSEUP.
- Arguments of type `__m256` are split into four eightbyte chunks. The least significant one belongs to class SSE and all the others to class SSEUP.
- Arguments of type `__m512` are split into eight eightbyte chunks. The least significant one belongs to class SSE and all the others to class SSEUP.

Passing Once arguments are classified, the registers get assigned (in left-to-right order) for passing as follows:

1. If the class is MEMORY, pass the argument on the stack at an address respecting the arguments alignment (which might be more than its natural alignement).
2. If the class is INTEGER, the next available register of the sequence `%rdi`, `%rsi`, `%rdx`, `%rcx`, `%r8` and `%r9` is used[19].
3. If the class is SSE, the next available vector register is used, the registers are taken in the order from `%xmm0` to `%xmm7`.
4. If the class is SSEUP, the eightbyte is passed in the next available eightbyte chunk of the last used vector register.
5. If the class is X87, X87UP or COMPLEX_X87, it is passed in memory.

When a value of type `_Bool` is returned or passed in a register or on the stack, bit 0 contains the truth value and bits 1 to 7 shall be zero[20].

Returning of Values The returning of values is done according to the following algorithm:

1. Classify the return type with the classification algorithm.

2. If the type has class MEMORY, then the caller provides space for the return value and passes the address of this storage in `%rdi` as if it were the first argument to the function. In effect, this address becomes a "hidden" first argument. This storage must not overlap any data visible to the callee through other names than this argument.

   On return `%rax` will contain the address that has been passed in by the caller in `%rdi`.

3. If the class is INTEGER, the next available register of the sequence `%rax, %rdx` is used.

4. If the class is SSE, the next available vector register of the sequence `%xmm0, %xmm1` is used.

5. If the class is SSEUP, the eightbyte is returned in the next available eightbyte chunk of the last used vector register.

The end of the input argument area shall be aligned on a 16 (32 or 64, if `__m256` or `__m512` is passed on stack) byte boundary. [11] In other words, the stack needs to be 16 (32 or 64) byte aligned immediately before the call instruction is executed.

Using syscall

1. User-level applications use as integer registers for passing the sequence `%rdi`, `%rsi`, `%rdx`, `%rcx`, `%r8` and `%r9`. The kernel interface uses `%rdi`, `%rsi`, `%rdx`, `%r10`, `%r8` and `%r9`.

2. A system-call is done via the `syscall` instruction. The kernel clobbers registers `%rcx` and `%r11` but preserves all other registers except `%rax`.

3. The number of the syscall has to be passed in register `%rax`.

4. System-calls are limited to six arguments, no argument is passed directly on the stack.

5. Returning from the `syscall`, register `%rax` contains the result of the system-call. A value in the range between -4095 and -1 indicates an error, it is `-errno`.

6. Only values of class INTEGER or class MEMORY are passed to the kernel.

Stack Frame

Position	Contents	Frame
`8n+16(%rbp)`	memory argument eightbyte n	Previous
	...	
`16(%rbp)`	memory argument eightbyte 0	
`8(%rbp)`	return address	Current
`0(%rbp)`	previous `%rbp` value	
`-8(%rbp)`	unspecified	
	...	
`0(%rsp)`	variable size	
`-128(%rsp)`	red zone	

Once registers are assigned, the arguments passed in memory are pushed on the stack in reversed (right-to-left[21]) order.

References

- Guide to x86 Assembly
- x86 Assembly/X86 Architecture - Wikibooks, open books for an open world
- ABI
- x86 and amd64 instruction reference
- NASM - The Netwide Assembler