Electron Binary Interface

x86_64 Architecture Supplement

Version 0.9.0

Madd Games

September 2016

Table of Contents

1. Introduction

This document specifies the binary interface for Electron on the *x86_64* (a.k.a. *AMD64*, *x64*) processor architecture. It is a supplement to the machine-independent *Electron Generic Binary Interface* (*EGBI*) specification and fills in the gaps which the EGBI specification defines as *architecture-specific*.

1.1. General Architecture Information

On the x86_64, the type `natural_t` is defined as `int64_t` and `unatural_t` is defined as `uint64_t`, as this is a 64-bit architecture. As normal, the `%rsp` register is used as the stack pointer, and the stack grows downwards. 128 bytes above the stack pointer are known as the *red zone* – asynchronous signals shall leave this area untouched, and functions may put arbitrary data in the red zone.

Before a `call`, the stack must be aligned on a 16-byte boundary. After a function returns, the stack must return to the value it was before issuing the call; that is, the caller cleans up the stack.

Frame information and arguments are allocated on the normal stack marked by `%rsp`.

2. Calling Convention

This section describes architecture-specific details of the calling convention defined in *EGBI Section 3. Calling Convention*.

Calls are issued using the `call` instruction. The argument array which goes into the *argument-register* is allocated on the stack.

2.1. Registers

The register mappings are as follows:

- The *argument-register* is `%rsi`.
- The *this-register* is `%rdx`.
- The *count-register* is `%rcx`.
- The *frame-info-register* is `%rbp`.
- The *return-register* is `%rax`.

The registers `%rsp`, `%rbp` and `%rbx` and `%r15` are non-volatile – their values must be restored by the callee and hence preserved across function calls. All other registers are volatile.

2.2. Frame Information

The EGBI allows for an architecture-defined part in the frame information structure. Hence, on the x86_64, we define the frame information structure as 32 bytes formatted as follows:

```
struct elec_frameinfo_x86_64
{
     struct elec_frameinfo             fiHeader;
     uint64_t                          fiPriv;
     void*                             fiRetAddr;
};
```

- `fiHeader` is the standard frame information structure as defined by the EGBI.
- `fiPriv` is private data; the function that owns this frame may put whatever it wants here, and other functions (or invocations of the same function) or the ABI are not allowed to change it.
- `fiRetAddr` is the return address – the address to jump to when the function returns. This is needed to figure out which instruction in the previous frame has called this function, so that it is possible to determine if it was within the scope of a `try` block, as described in *Section 3. Exception Handling*.

2.3. Prologue

To ensure asynchronous interrupt safety, the frame information structure must be set up before it is placed in `%rbp` (we use this fact in the unwinding algorithm described in *Section 3. Exception Handling*). The stack pointer must also point to the return address until `%rbp` is set.