А.В. Макаров, С.Ю. Скоробогатов, А.М. Чеповский

Учебный курс “CIL и системное программирование в Microsoft .NET”

Лекция 8.

Поток инструкций языка CIL

Введение

- Язык CIL (Common Intermediate Language) является независимым от аппаратной платформы объектно-ориентированным ассемблером, используемым на платформе .NET для представления исполняемого кода
- Инструкции CIL можно разделить на четыре основные группы:
  - инструкции общего назначения
  - инструкции для работы с объектной моделью
  - инструкции генерации и обработки исключений
  - неверифицируемые инструкции

8.1. Формат потока инструкций

- Тело метода в сборке .NET закодировано в виде потока инструкций языка CIL
- Поток инструкций представляет собой массив байт, в котором размещены последовательности байт, кодирующие каждую инструкцию
- Инструкции размещаются последовательно друг за другом без промежутков
- Поток инструкций CIL предназначен для JIT-компиляции и не может содержать “неправильных” последовательностей байт

8.1.1. Формат инструкции

- Последовательность байт, кодирующая инструкцию CIL, начинается с кода инструкции
  - Часто используемые инструкции имеют однобайтовые коды. Инструкции, которые используются реже, имеют двухбайтовые коды (при этом первый байт всегда равен 0xFE)
- Многие инструкции имеют дополнительные встроенные операнды (inline operands), которые находятся прямо в потоке инструкций
- Встроенные операнды размещаются в потоке инструкций сразу после кода инструкции

Варианты встроенных операндов инструкций CIL

Операнд	Размер в байтах	Описание
none	0	У некоторых инструкций встроенные операнды отсутствуют.
int8	1	Знаковое 8-битовое целое число
int32	4	Знаковое 32-битовое целое число
int64	8	Знаковое 64-битовое целое число
unsigned int8	1	Беззнаковое 8-битовое целое число
unsigned int16	2	Беззнаковое 16-битовое целое число
float32	4	32-битовое число с плавающей запятой
float64	8	64-битовое число с плавающей запятой
token	4	Токен метаданных
switch	переменный	Массив адресов переходов для инструкции **`switch`**

Кодирование встроенного операнда инструкции **switch**

- Инструкция **switch** осуществляет множественный условный переход в зависимости от некоторого целого значения, которое берется из стека вычислений
- Ее встроенный операнд представляет собой массив адресов переходов, кодируемый следующим образом:
  - Сначала идет 32-разрядное целое число без знака, обозначающее количество адресов переходов (размер массива)
  - Затем следуют сами адреса. При этом каждый адрес кодируется в виде 32-разрядного целого числа со знаком

Примеры кодирования инструкций CIL

- Инструкция **ldarg**.0 загружает на стек вычислений значение первого аргумента метода. Она является сокращенной версией инструкции **ldarg**, не содержит встроенных операндов и имеет код 0x02:

  /* 02 */ **ldarg**.0
- Инструкция **arglist** загружает на стек вычислений специальный описатель массива переменных параметров метода. Она не содержит встроенных операндов и имеет двухбайтовый код 0xFE 0x00:

  /* FE 00 */ **arglist**
- Инструкция **ldc**.i4.s 16 загружает на стек вычислений целочисленную константу 16. Она является сокращенной версией инструкции **ldc**.i4, имеет код 0x1F и содержит встроенный операнд типа int8:

  /* 1F | 10 */ **ldc**.i4.s 16

Примеры кодирования инструкций CIL (часть 2)

- Инструкция **ldc**.r4 1.0 загружает на стек вычисления число 1.0 (константу с плавающей запятой). Она имеет код 0x22 и содержит встроенный операнд типа float32:

  /* 22 | 0000803F */ **ldc**.r4 1.0
- Инструкция **isinst** System.String служит для динамической проверки типа объекта на стеке вычислений. Она имеет код 0x75 и содержит встроенный операнд типа token, в котором хранится токен метаданных, указывающий на тип:

  /* 75 | (02)00000F */ **isinst** System.String
  - В скобки помещен первый байт токена метаданных, обозначающий номер таблицы метаданных. Обратите внимание, что значение токена метаданных для типа System.String в различных сборках может отличаться.
- Инструкция **call** System.String::Compare вызывает метод. Ее встроенный операнд содержит токен метаданных, указывающий на описание вызываемого метода:

  /* 28 | (06)0000CD */ **call** System.String::Compare

8.1.2. Адреса переходов

- В состав набора инструкций CIL входят инструкции для организации условных и безусловных переходов. Встроенные операнды этих инструкций содержат адреса переходов
- Абсолютный адрес инструкции - смещение первого байта инструкции относительно начала потока инструкций
- Цель перехода (branch target) - инструкция, на которую передается управление в результате выполнения инструкции перехода
- В качестве адресов перехода используются не абсолютные адреса целей перехода, а так называемые относительные адреса

Вычисление относительного адреса цели перехода

```
                ...
target_addr:    add             ; цель перехода
                ...
                br  rel_addr    ; инструкция перехода
next_addr:      ...
```

- Относительный адрес является разностью абсолютного адреса цели перехода и абсолютного адреса инструкции, непосредственно следующей за инструкцией перехода:

  reladdr := target_addr - next_addr

- Допустимы только такие адреса, которые указывают на первые байты инструкций в теле данного метода

8.1.3. Ограничения на последовательности инструкций

- В спецификации языка CIL в описании каждой инструкции указаны условия, при которых допустимо ее использование
- Кроме того, на формирование последовательности инструкций наложен ряд ограничений, позволяющих упростить создание JIT-компилятора
- В программах допускаются не любые сочетания инструкций, а только те сочетания, которые удовлетворяют некоторым условиям

Ограничение на структуру стека вычислений

- Структура стека вычислений определяется количеством и типами значений, лежащих на стеке вычислений.
- Для любой инструкции, входящей в поток инструкций, структура стека должна быть постоянной вне зависимости от того, из какого места программы на нее передается управление

Ограничение на размер стека вычислений

- В заголовке метода должна быть указана максимальная глубина стека вычислений
- Цель введения этого ограничения состоит в том, чтобы JIT-компилятор, приступая к компиляции метода, мог сразу выделить нужное количество памяти под свои внутренние структуры данных

Ограничение на обратные переходы

- Если при последовательном переборе инструкций, формирующих тело метода, JIT-компилятор встречает инструкцию, расположенную сразу за инструкцией безусловного перехода, и если на эту инструкцию еще не было перехода, то JIT-компилятор предполагает, что стек для этой инструкции должен быть пуст

```
    ...
    br  L2
L1: ldc.0    ; здесь стек считается пустым
    ...
L2: br  L1   ; обратный переход на L1
    ...
```