ОБЩИЕ ПОДХОДЫ К ПОСТРОЕНИЮ ЭМУЛЯТОРОВ X86-СОВМЕСТИМЫХ ПРОЦЕССОРОВ

Шпаковский А.П.

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Оношко Д.Е. – старший преподаватель

В настоящее время не теряет актуальности построение эмуляторов аппаратно-программных платформ, применение которых, с учетом современных реалий, проблематично или нецелесообразно из соображений, как правило, безопасности и удобства работы с памятью.

Одним из ключевых направлений деятельности разработчиков современных эмуляторов является поддержка процессоров архитектуры x86, работающих в реальном режиме[1]. Несмотря на то, что этот режим давно перестал быть основным режимом на современных процессорах, не стоит забывать о его предельной простоте, приходящейся весьма кстати для образовательных целей и знакомства и x86-архитектурой.

На момент разработки проекта можно выделить несколько программных средств, предоставляющих подобные возможности. Наиболее популярные из них - эмуляторы Bochs[3] и DOSBox[2], в центре внимания которых эмуляция процессоров архитектуры x86, в частности, их реального режима. В обоих случаях разработчики своей целью ставили поддержание максимально большого количества ПО для разных моделей процессоров и периферийных устройств. Обеспечение подобного рода гибкости в той или иной мере негативно сказывается на скорости работы программы ввиду невозможности применения одинаковых способов эмуляции и возможных оптимизаций для разных модификаций эмулируемого оборудования.

В разработанном прототипе эмулятора за основу взяты характеристики процессора Intel 8086, положившего начало архитектуре x86. Целевая платформа также использует обратно совместимую с x86 архитектуру и набор команд, что позволяет значительно оптимизировать некоторую алгоритмически реализуемую часть задач, переложив ее на реальное «железо».

Рассмотрим более подробно реализацию некоторых узлов системы и приемы эмуляции.

Регистры. В качестве эмулируемых регистров удобно использовать 16-битные участки памяти, над которыми выполняются те или иные операции.

Флаги. Как уже было сказано, ввиду схожести архитектуры гостевой и хостовой платформы и семантической схожести эмулируемых и аппаратно выполняющихся инструкций, допустимо использование некоторых флагов реального процессора. Достаточно считать их из регистра флагов после выполнения очередной инструкции до того, как они изменятся.

Достигается это копированием содержимого регистра флагов с помощью инструкций PUSHFD и POP. Первая инструкция помещает флаги на стек, откуда, с помощью второй команды, они переносятся в выбранный регистр или область памяти, где ими можно манипулировать с помощью битовых операций.

Разбор инструкций. Первостепенной задачей эмулятора, без которой не имеет смысла все вышеописанное, является разбор и выполнение инструкций, записанных во входном файле. Наиболее простым и удобным вариантом является применение массива указателей на процедуры, которыми эмулируются те или иные машинные команды. В документации Intel[4, Vol. 2C A-8] его принято представлять в виде матрицы 16×16. Необходимый указатель находится в строке с номером, соответствующим старшей тетраде первого байта текущей инструкции. Младшая тетрада задает, соответственно, положение в этой строке.

Самыми распространенными из эмулируемого набора инструкций являются команды,

которые, согласно современной документации Intel [4], принято называть инструкциями с ModR/M байтом. Для этого случая были разработаны несколько процедур, позволяющие разбирать такие инструкции одинаково, вне зависимости от типа операндов, ведь на хостовой системе и то, и другое является оперативной памятью.

Работает это так: исходя из того, что размер и тип операндов, которые может принимать инструкция известен заранее, вызывается одна из четырех возможных функций, работающих со своим «шаблоном». В теле функции на основании структуры ModR/M байта информация об операндах конкретизируется и, основываясь на уже точных данных, вычисляются адреса эмулируемых регистров или операндов в памяти. Эти вычисления, для удобства применения, также были вынесены в отдельные подпрограммы.

Прерывания. Не менее важной задачей является поддержка прерываний и, соответственно, функций гостевой системы. Здесь проблематично применить унифицированный подход для эмуляции всех прерываний, так как эмуляция реального режима производится на защищенном. Соответственно набор инструментов, к которым можно получить прямой доступ существенно ограничен и эмуляция некоторых аппаратных узлов достигается взаимодействием с интерфейсом хостовой ОС.

Как частный случай рассмотрим эмуляцию функции получения символа из стандартного потока ввода, реализуемую на ОС Windows. Дойдя до вызова этой функции цикл разбора и выполнения инструкций прерывается, а адрес ожидающей выполнения инструкции сохраняется. От окна ожидается сообщение WM_CHAR [5]. После его получения в регистр помещается код считанного символа и цикл разбора инструкций запускается заново, имея на входе адрес инструкции, до которой еще не дошло выполнение.

Таким образом, в результате проделанной работы был реализован набор приемов эмуляции основных узлов процессора архитектуры x86, достаточный для выполнения некоторых программ, подразумевающих работу в реальном режиме. Весь набор реализованных инструментов удалось разбить на логически обособленные подпрограммы. Получившиеся «блоки» можно применять отдельно от остальных или изменять один без внесения правок в остальные, а также добавлять новые, расширяя возможности эмулятора.

Список использованных источников:

1. Real mode – Wikipedia [Электронный ресурс]. - Режим доступа: https://en.wikipedia.org/wiki/Real_mode. Дата доступа: 10.12.2019

2. DOSBox v0.74-3 Manual [Электронный ресурс].—Режим доступа: https://www.dosbox.com/DOSBoxManual.html - Дата доступа: 06.02.2020 г.

3. Bochs Developers Guide [Электронный ресурс].—Режим доступа: http://bochs.sourceforge.net/doc/docbook/development/index.html - Дата доступа: 04.02.2020 г.

4. Intel 64 and IA-32 Architectures Software Developer's Manual / Intel Corporation, 2014

5. WM_CHAR message - Microsoft Software Developer's Network [Электронный ресурс].—Режим доступа: https://docs.microsoft.com/en-us/windows/win32/inputdev/wm-char — Дата доступа: 12.12.2019 г.