Ассемблер — Lurkmore

Народ требует ~~хлеба и~~ зрелищ!

Народ требует иллюстраций к статье!
В конце концов, если бы мы хотели почитать, мы бы пошли в библиотеку.

«Нам не дано предугадать, как слово наше отзовется… »

— Ф. И. Тютчев программирует на Ассемблере

Ассемблер или **asm** (англ. *assembler* — сборщик) — утилита, транслирующая исходники программы на *языке ассемблера* собственно в машинный код, то есть на язык бездушной машины. Для обратного превращения существует **дизассемблер** (англ. *disassembler* — разборщик, ломалка), широко используемый хакерами. Но это для шибко грамотных, а все остальные словом *«ассемблер»* называют сам **язык ассемблера** — простейший способ записи машинного кода с помощью расовых английских сокращений, называемых мнемониками (см. в конце статьи — если осилите). Поскольку язык ассемблера привязан к устройству процессора, программы на нем не являются переносимыми на иную компьютерную архитектуру, так как тип процессора жёстко определяет набор доступных команд машинного языка. Но это не так уж и важно, так как нет процессора кроме x86 и фон Нейман пророк его, а всякие Жабы и дотнеты вообще используют собственные виртуальные машины с собственным асмом — похуй, что в тысячу раз медленнее, зато переносимость и безопасность, блджад! Также, используя макросы, юный друг сможет убедиться в отсутствии принципиальной разницы между асмом и всеми остальными «православными» языками.

История

Когда компьютеры были большими как динозавры, а объем мозгов у них был примерно такой же, как у этих рептилий, и даже меньше, единственным способом программирования было забивание программы в память ЭВМ непосредственно в цифровом виде (иногда даже с помощью щелканья переключателями). Затем человек начал совершенствовать эту технику, постепенно перекладывая труд по генерации кода на плечи самой машины. Ибо заставлять высококвалифицированного человека перебирать биты и помнить кучу шестнадцатеричных (а то и восьмеричных) кодов обходится весьма дорого, как в деньгах, так и во времени.

Простейший транслятор (ассемблер первого поколения) позволял фактически просто писать машинные команды на «человеческом» языке, что позволило программерам немного расслабиться. Достаточно быстро туда добавились многочисленные свистелки и перделки, нарушающие Истинную Красоту Программирования На Ассемблере, но еще больше экономящие время и силы (а значит и деньги), как то макросы, библиотеки и прочее. Затем появились языки высокого уровня и компиляторы (более интеллектуальные генераторы кода с более понятного человеку языка) и интерпретаторы (исполнители человекописанной программы на лету). Которые совершенствовались, совершенствовались, совершенствовались, и досовершенствовались до «программирования мышкой», а компиляторы научились генерировать код, которому по скорости написанный человеком начал сливать[1]. Такие дела.

В целом история языков программирования протекает в направлении от программирования на языке компьютера до манипуляции абстракциями вроде `Послать.Лесом(всех(кому не нравится эта статья))` Ну или `(послать (лесом (всех (кому-не-нравится "эта-статья"))))` или

```
послать :: Куда -> [Люди] -> String
послать куда = map (\x -> (show x) ++ ", иди " ++ (show куда))
кванторВкуса :: (ОбъектЧувства a) => ТипКвантора -> ТипОтношения -> a -> [Люди]

посылаем :: String
посылаем =  let быдло = кванторВкуса Все НеНравится этаСтатья
            in послать быдло
```

На языке ассемблера этот код занял бы более 9000 строк. И потребовал бы долгой и кропотливой работы по своему созданию. Зато, в отличие от Windows, оно бы работало, а не только свистело, пердело, выдавало красивые иконки и жрало 640K памяти.

Асмофаги и студенты из Политеха

Среди программистов с длинной бородой, видевших еще те компьютеры-динозавры, есть такие, которые очень трепетно относятся к напейсанию кода, и считают, что ассемблер — это труЪ, а языки высокого уровня — шлак и унылое говно, предназначенные исключительно для быдлокодерства. При упоминании факта, что современные компиляторы генерируют гораздо более быстрый код, начинают срать кирпичами и обзываться нехорошими словами, ибо сама мысль, что машина может быть умнее человека, считается

недопустимой и кощунственной. При этом упускается тот факт, что компиляторы, вообще-то, пишут люди. Степень владения языком ассемблера считается мерилом отличия программиста от быдлокодера. Наиболее тяжелый клинический случай представляет молочный брат Линуса расовый чухонец Вилле Турьянмаа, написавший целую ОС — MenuetOS на чистом ассемблере. На ассемблере же была написана первая версия UNIX, коя им и не является по нынешним понятиям, так как позже была переписана чуть более, чем наполовину на C и помещена в анналы истории.

Тем не менее, программирование на языке ассемблера очень часто используется при написании программ, использующих возможности процессора, не реализуемые языками высокого уровня, а также программировании всевозможных нестандартных программистских хитростей. Драйвера для новых устройств тоже содержат в себе чуть менее, чем половину ассемблерного кода, коий при доведении их до ума ушастыми мартышками заменяют на стандартный C/C++. Часто компиляторы (чуть менее, чем все) предоставляют возможность вставлять ассемблерный код в текст программы.

Алсо, ассемблер используется для программирования всевозможных микроконтроллеров и сигнальных процессоров, параллельно с C-компиляторами. В этом случае зачастую важна компактность кода и скорость работы, которых компилятор может не обеспечить.

Понятно, что писать что-либо крупное на ассемблере в современном мире не будет даже самый отъявленный маньяк, даже большинство фирмваря и прочих глюкалок для микроконтроллеров сейчас пишется на чем-то более продвинутом. Но когда стоимость каждого лишнего не то что мегабайта, а полубайта в ОЗУ становится критичной (ибо их просто нет, например на каком-нибудь запущенном много лет назад спутнике, до которого можно достучаться по радио, но никак физически), выливаются в миллиарды и даже жизни (встроенный софт бортовых систем в авиации и космонавтике) — ассемблер strikes back! К примеру, софт для кораблей «Союз ТМА-М», «Прогресс М-М» написан на Си (по крайней мере, российский сегмент МКС). ПО ЦВМов «Бурана» писалось на ПРОЛ-2. «Союз ТМА» — не на Си, у него БЦВК «Аргон-16», сначала программировали его на ассемблере, в поздних машинах транслировали на высокоуровневый язык. Спутники «Ямал» и «Белка» написаны на Си.

Ассемблер и Линукс

Хотя программирование на ассемблере в Unix-мире считается моветоном, поскольку нарушает один из основополагающих принципов unix way «жертвуй производительностью ради переносимости», даже тут находится место разногласиям, способным вылиться в настоящую драму.

Дело в том, что встроенный ассемблер большинства Unix-ов (и Linux в том числе) использует синтаксис (AT&T-синтаксис), в корне отличающийся от оригинального Intel-синтаксиса. А поскольку Intel-синтаксис у большинства труЪ-линуксоидов прочно ассоциируется с маздаем, он объявляется ересью, и всем, кто пользуется nasm, yasm или fasm пророчатся вечные муки, страшный суд, ад и погибель, хотя gas тоже владеет интеловским синтаксисом. Правда все ELF-инфекторы всё же приходится писать на старом добром асме.

Алсо какой либо годной документации по ассемблеру AT&T в интернете хрен найдешь. По этой причине программистов на AT&T крайне мало. А по этой причине некому писать документацию.

Ассемблер в кино


Оно.

Как минимум один раз снялся в кино, но зато в каком! В первой части Терминатора можно видеть как Киборгъ предпочитает MOS Technology 6510/8500 ассемблер.

Необходимое пояснение. На скриншоте видна программка из чередующихся инструкций LDA-STA-LDA-STA...(load-store-load-store). В семействе 6502 программы состоят чуть менее, чем полностью из LDA/LDY/LDX/STA/STX/STY вследствие наличия всего 3х 8-битных регистров. Наличие почти полного отсутствия регистров компенсируется нулевой страницей памяти, откуда-куда постоянно приходится перекладывать байтики, так как хранить их более негде. В данном ЦПУ 13 режимов адресации на всего 53 команды. Алсо порты ввода-вывода замаплены в адреса памяти, так что чтение-запись из-в порты осуществляется также этими командами. Короче: программы в Терминаторе имеют вполне осмысленный вид и не являются кучей команд от балды. Подробнее: **Этот удивительный Терминатор**[/48076].

Во второй части интерфейс Терминатора более очеловечен. Впрочем, это можно объяснить тем, что ПО для первого терминатора делал Скайнет, а для второго набыдлокодил Джон Коннор.

Алсо, ассемблер семейства 8080 (у нас серия КР580) засветился в давнем аниме «Megazone 23».

Алсо, в качестве исходника чат-бота Киса в фильме }{0TT@БЬ)Ч был показан исходник чат-бота Eliza на асме для калькулятора TI-83 Plus с процессором Z80.

Алсо, в фильме «Элизиум» (2013) пафосный топ-манагер кодит на асме.

Область применения

- Популярен для допиливания зацикливаемых кусков программы… в роли **напильника**. Перепиливание критических участков кода может принести PROFIT, а может и не принести. В любом случае, заниматься таким перепиливанием **стоит только тогда, когда у вас на руках уже полностью работающий алгоритм**, который можно было бы ускорить, а не наоборот.
- Для использования в софтине новых команд, доступных в новых процессорах. Компилятор хоть и оптимизирует код высокого уровня при компиляции, но… практически никогда не способен генерировать инструкции из расширенных наборов команд типа AVX, CTV, XOP и т. п. Потому что команды в процессоры добавляют быстрее, чем в компиляторах появляется логика для генерации этих команд.
- Используется для написания кода, создание которого невозможно (или затруднено) на языках высокого уровня, например, получение дампа памяти/стека. Даже когда аналог на языке высокого уровня возможен, профит от языка ассемблера может быть значительным в разы. Например, реализация подсчета среднего арифметического двух чисел с учётом переполнения для x86 процессоров занимает всего 2 команды (сложение с выставлением флага переноса и сдвиг с заёмом этого флага). Аналог на языке высокого уровня ((long) x + y) >> 1
  1. может не работать в принципе, ведь sizeof(long) == sizeof(int). Ну или может бомбануть, как это было на ракете Ариан-5 (64-ичные переменные с плавающей запятой float преобразовывались в 16 бит фиксированных, когда программа дошла до 32767+1, у ракеты включилась система «самоуничтожение на случай попадания в мир КГБ» прямо на космодроме.
  2. при компиляции конвертируется в чуть менее чем дохрена команд процессора.

Ещё пример: перемножение чисел с фиксированной запятой в языке, в который не вшита поддержка такого формата чисел. На ассемблере это умножение eax на edx с сохранением 64-битного результата в edx:eax с последующим сдвигом вправо (shrd). На ЯВУ это выглядит либо как (x*y)>>16, что не даёт верный результат при выходе результата умножения за 32 бита, либо как (int64(x)*y)>>16, что конвертируется в дохрена команд. Впрочем, в данных примерах современные компиляторы умеют правильно понимать намерения программиста и генерировать оптимальный код.

- Также может использоваться задротами для написания очень компактных программ. Пример: клиент для аськи, занимающий в скомпилированном виде 34 КБ. Другой пример: тоже IM-клиент Faim 0.21 — 12.61 КБ. Отдельная дисциплина этой специальной олимпиады — демки.
- В вузах изучается для вливания в МНУ студентоты познаний об устройстве и работе компьютера на уровне повыше кликанья мышкой по окошкам. Отличное лекарство от быдлокодерства (ну, а для оказавшихся неспособными осилить, соответственно — окончательное решение).
- Является единственным языком программирования для создания достойных инфекторов — Cih, Sality, Sinowal.
- Чуть более чем половина программ для GPU пишутся на ассемблере, наряду с использованием языков высокого уровня HLSL или GLSL. Хороший, годный шейдер обязательно содержит асму, ибо GPU — это принципиально узкое место для выделения графической составляющей в монитор (Есть, конечно и более современные методы типа использования OpenCL, но всё же ASM << C).
- Оптимизация невъебенно затратных математических алгоритмов. Часто встречается в исходниках фильтров для так любимого видеопиратами и аниматорами винрарнейшего AviSynth.
- Промежуточный этап между неким левым синтаксисом (не обязательно даже языком, это может быть визуальный «конструктор» программы вроде ДРАКОНа) и исполнимым кодом. Петя хочет анализ Фурье на ЦСП на участке в миллисекунду, а Вася в две? Нет проблем, нажимайте кнопочки и собирайте в проект полученные пятидесятиметровые ассемблерные сырцы, работающие с предельно возможной именно для конкретного случая скоростью. Не компилятор же отдельный писать ради Васи и Пети, честное слово.
- Программирование под устаревшие платформы (например, NES или ZX Spectrum), которым мешает использовать «сбалансированный» Си крайне низкая производительность и нужда в хитрых изъёбах при обработке данных и графики.
  Тамошний ассемблер, в отличие от этих ваших x86, обладает Духовностью и Минимализмом.

Пример программирования

Нижеследующий кусок кода переключает процессор из реального (16-битного) режима в длинный (64-битный). Эпичность заключается в «прыжке через десятилетия», поскольку переключает ЦП x86 из режима совместимости со своим предком 1979 года рождения, в коем он находится сразу после подачи питания, в современный (2003 года разработки), где доступны все инструкции и всё адресное пространство памяти.

Данный пример написан для ассемблера nasm:

```
BITS 16
                ; Тут шла длинная инициализация системных таблиц; прерывания запрещены, NMI блокировано

                lidt [idtr_image]          ; IDTR указывает на 64-битную таблицу прерываний
                lgdt [gdtr_image]          ; GDTR указывает на глобальную дескрипторную таблицу
                mov eax,cr4                ; Установим CR4.PAE=1
```

```
                or ax,20h
                mov cr4,eax

                mov ecx,0C0000080h          ; Установим EFER.LME=1
                rdmsr
                or ax,100h
                wrmsr

                mov eax,[pml4_base]         ; Установим CR3 на корневую таблицу страниц
                mov cr3,eax

                mov eax,cr0                 ; Все готово. Установкой CR0.PE и CR0.PG включаем длинный режим
                or eax,80000001h
                mov cr0,eax

                db 66h                      ; Это переводится как jmp far 0008:start_64
                db 0eah                     ; Вписываем цыфирками, поскольку надо наставить нужных префиксов
                dd start_64                 ; Чистим очередь команд и прыгаем в 64-битный код
                dw 08h
BITS 64
start_64:       mov rsp,STACK_BASE          ; установим стек
               ; Здесь хуярим 64-битный код...
```

Вообще, в среде ASM программистов существует целая война ассемблеров. Кто-то предпочитает nasm, кто-то yasm, tasm, masm или даже fasm. На самом деле вроде бы и ерундовая вещь, но каждый ассемблер имеет свои особенности и целый ряд анальных ограничений. Поэтому не удивляйтесь, если кто-то будет с пеной у рта отстаивать свой ассемблер, или тому, что код, написанный под одну разновидность, на другой попросту не оттранслируется. Ассемблер имеет множество диалектов, некоторые из них довольно продвинуты, например, никому не нужные макросы в fasm чем-то напоминают язык высокого уровня, поэтому он вызывает в среде ассемблерщиков некоторый баттхёрт, да ещё и ассоциации с хорошо оптимизированным и разбитым на отдельные операции с прямыми указателями Си делают кон**троль**ный выстрел.

Мнение изучающей студентоты

На самом деле, соль Ассемблера в том, что он делает ВСЁ, кроме того, что нужно. Термин «Научиться программировать на Ассемблере» (см. ассемблировать, «оседлать», «обуздать», «поработить» Ассемблер) подразумевает вывод на протяжении долгого и попабольного обучения хитроумного плана, который позволит наебать Ассемблер незаметно для него, тем самым заставив его работать на вас. Судя по всему, успеха добиваются единицы. Остальные же, которые, не понимая того, становятся рабами Вышеназванного, теряют волю, сходят с ума и, в лучшем случае, попадают в армию.

Мнение изучившей студентоты

Будучи изученным в течение нескольких лет в вузе и применяемым на практике, добавляет стопицот к ЧСВ, что мешает при необходимости программировать на не-ТруЪ-языках при смене работы (а также пола, возраста, профессии и т.п). Например, писать на javascript после ассемблера походит на процесс создания красных сердечек на торте из сахара паяльником: дорого, долго и болезненно. Зато чем-то похоже на ТруЪ

Алсо

- Эрик Дрекслер придумал и популяризовал концепт особого наноустройства — ассемблера. Суть™ девайса состоит в том, что продукты своей деятельности он собирает из отдельных атомов. Размножается так же. Будучи в больших количествах, как джинн из бутылки, сможет лечить людей, перерабатывать любой мусор, производить изделия безупречного качества и выполнять всяческие прихоти. Однако, выйдя из-под контроля, способен учинить тотальную демократию, превратив всю поверхность Земли вместе со всем ее содержимым в идеальное высокотехнологическое говно. Визуально винрарно такой сценарий показан в сериале Stargate SG-1. Местные репликаторы превратили целую планету в себя и только тогда задумались, зачем оно им было надо.
- Как упомянуто в заголовке, в переводе с расово-пиндосского **Assembler** означает «сборщик». То есть в тех же СШП ассемблером может называться всё (неживое и живое), что занимается сборкой чего-либо.
- TASM — Tomahawk Anti-Ship Missile (BGM-109B TASM). Пиндосская противокорабельная крылатая ракета «Томагавк».
- В запланированной небезызвестным Нотчем игре — 0x10c — нужно было бы иметь знания Assembler'a для того, чтобы твой корабль мог сам по себе ремонтироваться, поровну распределять энергию между системами и приносить тебе удовольствия самоудовлетворения, пока ты, дорогой быдлокодер, фиксишь косяки этого самого Assembler'a. Но после полутора лет разработки Нотч слился, так и не осилив сабж. Так-то!

Галерея

См. также

- Pascal
- PHP
- C++
- ++i + ++i

Ссылки

- Довольно большой ресурс, посвященный ассемблеру
- Жожошник проверяет, а так уж ли круты компилёры по сравнению с Вышеназванным?
- TASM Visual — IDE для компилятора Borland Turbo Assembler

Примечания

1. ↑ Речь идёт только о последнем эшелоне оптимизации — оптимальном генерировании машинных операций. А в наше время узкое место и так давно уже не в нём, а в бездарной архитектуре комплексов ПО в целом, из-за чего машина занимается переливанием из пустого в порожнее — но с предельно оптимизированной скоростью струи, да — это в первую очередь. А еще очень негативно на скорости сказывается незнание быдлокодерами основ теории алгоритмов и структур данных. Второй пункт в последнее время нивелируется невъебенно навороченными стандартными (или специальными) библиотеками языков и сред разработки, что в конечном итоге способствует терминальному отупению тех же быдлокодеров и возвратом к пункту 1 этой ссылки

Software
12309 1C 3DS MAX 8-bit Ache666 Alt+F4 Android BonziBuddy BrainFuck BSOD C++ Chaos Constructions Cookies Copyright Ctrl+Alt+Del Denuvo DOS DRM Embrace, extend and extinguish FL Studio Flash FreeBSD GIMP GNU Emacs Google Google Earth I2P Internet Explorer Java Lolifox LovinGOD Low Orbit Ion Cannon Me MediaGet MenuetOS Microsoft Miranda Movie Maker MS Paint Open source Opera PowerPoint PunkBuster QIP Quit ReactOS Rm -rf SAP SecuROM Sheep.exe Skype StarForce Steam T9 Tor Vi Windows Windows 7 Windows Phone 7 Windows Phone 8 Windows Vista Wine Winlogon.exe Wishmaster Word ^H ^W Автоответчик Антивирус Ассемблер Баг Билл Гейтс и Стив Джобс Блокнот Бот Ботнет Браузер Варез Винлок Вирусная сцена Генерал Фейлор Глюк Гуй Даунгрейд Демосцена Джоэл Спольски Донат Защита от дурака Звонилка Интернеты Кевин Митник Китайские пингвины Костыль Красноглазики Леннарт Поттеринг Линуксоид Линус Торвальдс Лог Ман Машинный перевод Мегапиксель

Языки программирования
++i + ++i 1C AJAX BrainFuck C Sharp C++ Dummy mode Erlang Forth FUBAR God is real, unless explicitly declared as integer GOTO Haskell Ifconfig Java JavaScript LISP My other car Oracle Pascal Perl PHP Prolog Pure C Python RegExp Reverse Engineering Ruby SAP SICP Tcl TeX Xyzzy Анти-паттерн Ассемблер Быдлокодер Выстрелить себе в ногу Грязный хак Дискета ЕГГОГ Индусский код Инжалид дежице

Капча КОИ-8 Костыль Лог Метод научного тыка Очередь Помолясь Проблема 2000 Программист Процент эс Рекурсия Свистелки и перделки Спортивное программирование СУБД Тестировщик Умение разбираться в чужом коде Фаза Луны Фортран Хакер Языки программирования

w:Ассемблер en.w:Assembly language ae:Machine Code