Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «**Информатика и системы управления**»

КАФЕДРА «**Программное обеспечение ЭВМ и информационные технологии**»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

НА ТЕМУ:

«Брух брах помогите((()»

Студент группы **<ИУ7-83Б>**		**<М. Ю. Нитенко>**
	(Подпись, дата)	(И.О. Фамилия)
Руководитель ВКР		**<А. А. Оленев>**
	(Подпись, дата)	(И.О. Фамилия)
Нормоконтролер		**<Это кто>**
	(Подпись, дата)	(И.О. Фамилия)

***2022** г.*

РЕФЕРАТ

Расчетно-пояснительная записка 19 с., 1 рис., 0 табл., X ист., X прил.

КЛЮЧЕВЫЕ СЛОВА

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

ВВЕДЕНИЕ

Трансляция кода x86 -> ARM.

Процессоры архитектуры ARM занимают большую долю рынка, еще в 2015 году они составляли 35% от рынка процессоров, однако в основном они использовались в портативных устройствах [1]. С появлением процессоров M1 от компании Apple большое число людей начало пользоваться компьютерами на основе архитектуры ARM в домашней обстановке (типа personal computers). Однако, программы собранные под архитектуру x86 не смогут работать на таких компьютерах, им необходим или транслятор, такой как Rosetta 2, или виртуальная машина поддерживающая необходимую архитектуру.

Таким образом что? Боремся с Rosetta 2 что ли? Чето заставило задуматься... Ну пока просто обзор стратегий по оптимизации онлайн и офлайн трансляции кода x86 -> ARM.

1 Аналитическая часть

Проблемы эмулятора:

— управление кэшем транслированного кода;

— выделение регистров;

— оптимизация условных блоков;

— direct block chaining????? lmao;

— управление памятью;

— поддержка самоизменяемого кода;

— поддержка исключений;

— поддержка аппаратных прерываний;

— эмуляция режима пользователя.

1.1 QEMU

(инфа 2005 года...) В QEMU реализована динамическая трансляция инструкций. QEMU транслирует команды при помощи кода заранее сгенерированного при помощи GCC.

Каждая инструкция процессора разбивается на микрооперации, эти микрооперации реализованы на языке C. Микрооперации выбираются так, чтобы их количество было много меньше количества всех возможных комбинаций инструкций и операндов процессора.

Программа входящая в QEMU — dyngen использует объектные файлы с микроинструкциями и генерирует на их основе динамический генератор кода, именно он используется во время выполнения программы для трансляции. [2]

1.1.1 Оптимизации используемые в QEMU

Главной идеей является возможность передачи константных параметров микрооперациям. Для этого GCC генерирует специальный код (?) для каждых константных параметров (я щяс умру..).

Например:

movl_T0_r1 # T0 = r1

addl_T0_im -16 # T0 = T0 - 16

movl_r1_T0 # r1 = T0

Таким образом уменьшается количество необходимых микроопераций, так как можно реализовать операции загрузки всех необходимых регистров во временные регистры и проводить операции именно с ними. Эти регистры обычно хранятся в регистрах хоста (например T0, T1 и T2 хранились бы в rax, rbx, rcx на x86).

Важным условием является наличие только одной точки выхода из микрооперации, действительно, наличие нескольких точек выхода сделает невозможным простое (без вызова процедуры) соединение микроинструкций.

Трансляция кода происходит блоками, блоком является часть кода в которой состояние процессора можно определить на этапе трансляции. QEMU транслирует весь такой блок до условного jump или иной инструкции изменяющей состояние процессора.

QEMU (в 2005, проверить что щяс (потому что типа хотели динамически выделять временные регистры и экономить время, о как!)) использует статическое выделение регистров. Каждый регистр эмулируемого процессора отображается на регистр хоста или на участок памяти.

ОПТИМИЗАЦИЯ ИФОВ! Главным замедляющим фактором при эмуляции является эмуляция условных операций (регистр eflags на x86). QEMU не обновляет условные регистры после каждой операции, вместо этого хранится один операнд (CC_SRC), результат (CC_DST) и тип операции (CC_OP). Например для 32-битного сложения $R = A + B$ хранится будут:

```
CC_SRC=A
CC_DST=R
CC_OP=CC_OP_ADDL
```

Этих данных достаточно чтобы восстановить B и нужные флаги, такие как ZF, SF, CF, OF.

После трансляции блока так же проверяется, используются ли переменные CC_SRC, CC_DST, CC_OP, если они не используются то им ничего не следует присваивать на стадии выполнения.

После выполнения блока QEMU ищет следующий блок в хэш-таблице, если он уже транслирован, начинается его исполнение, иначе запускается процесс трансляции. Если значение PC (program counter) известно заранее, QEMU может модифицировать блоки так, чтобы необходимый блок запускался сразу после предыдущего.

1.2 bruh

Список:

— первое;

— второе;

— пятое;

— десятое.

Формула:

$$c^2 = a^2 + b^2 \tag{1}$$

Ссылаемся на рисунок 1. Информация из источника [**?**].

Листинг 1: Пример кода

```
package main

import (
    "bufio"
    "fmt"
    "net/http"
)

func main() {
    resp, err := http.Get("http://gobyexample.com")
```

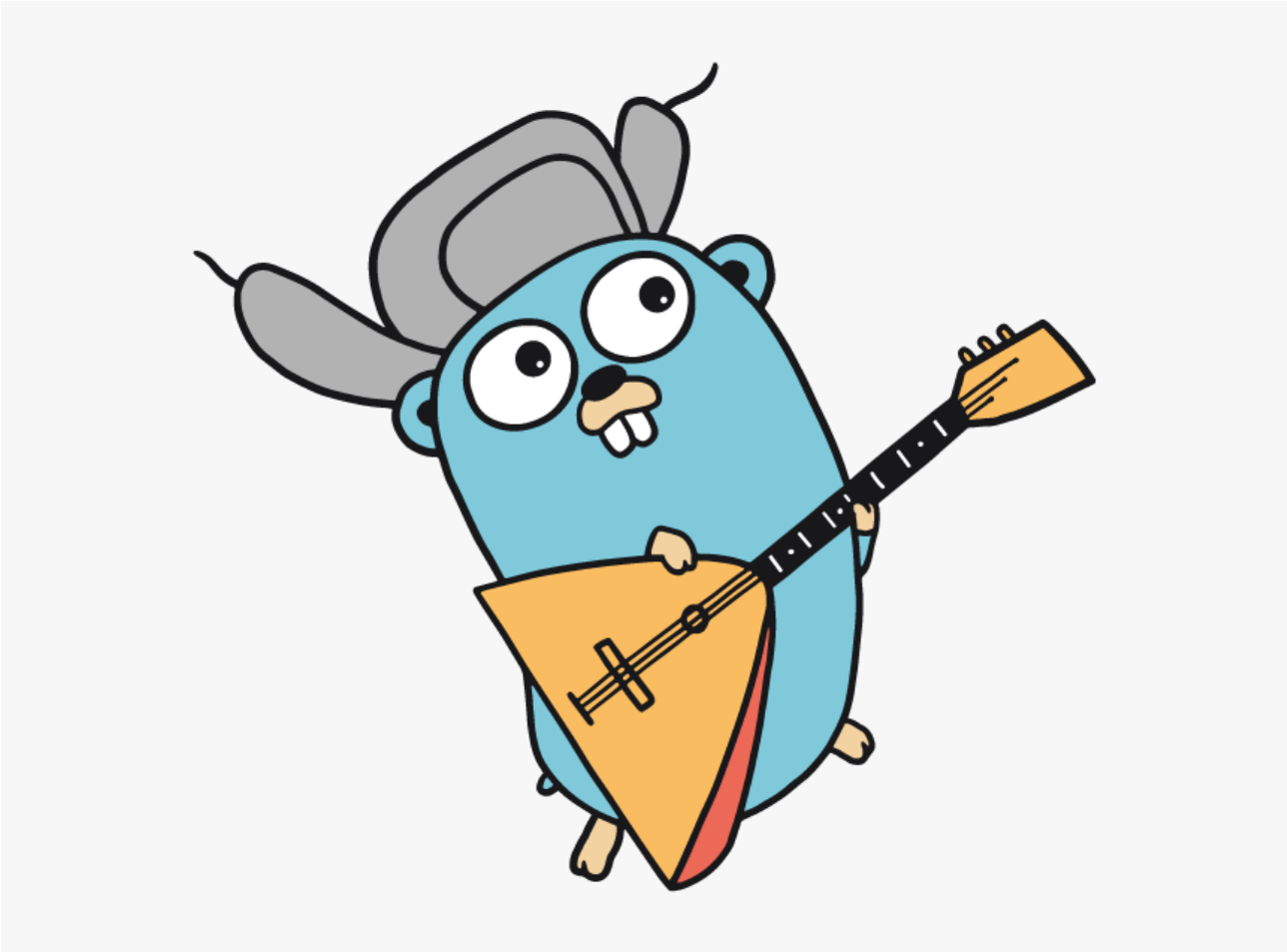

Рисунок 1 – Пример рисунка

```
11        if err != nil {
12            panic(err)
13        }
14        defer resp.Body.Close()
15
16        fmt.Println("Response status:", resp.Status)
17
18        scanner := bufio.NewScanner(resp.Body)
19        for i := 0; scanner.Scan() && i < 5; i++ {
20            fmt.Println(scanner.Text())
21        }
22
23        if err := scanner.Err(); err != nil {
24            panic(err)
25        }
26    }
```

2 Конструкторская часть

3 Технологическая часть

4 Исследовательская часть

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Annual Report 2015: Strategic Report [Электронный ресурс]. – Режим доступа: https://media.corporate-ir.net/media_files/IROL/19/197211/2016CustomWork/ARM_Strategic_Report.pdf, свободный – (24.11.2021) (преза арма как оформить??)

2. Bellard F. QEMU, a Fast and Portable Dynamic Translator [Текст] / Bellard F. // FREENIX Track: 2005 USENIX Annual Technical Conference. – 2005. – С. 41-42.

ПРИЛОЖЕНИЕ А