МультиОберон для ARM

Дагаев Дмитрий Викторович dvdagaev@oberon.org

Компилятор МультиОберон 1.2

МультиОберон это компилятор языка Оберон с 3 различными бэкендами:

- Генератором нативного кода x86 для системы BlackBox (1.7 с частичной поддержкой 1.6);
- Транслятором Ofront в язык C;
- Генератором кода LLVM.

МультиОберон это кроссплатформенный компилятор с поддержкой:

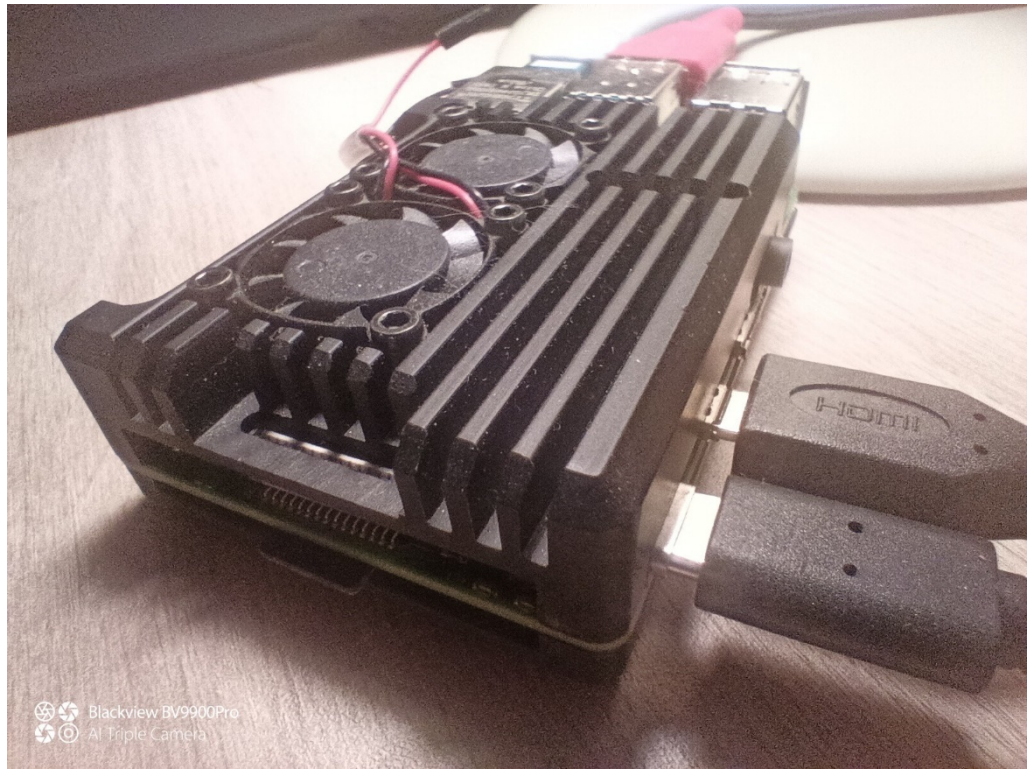
- Windows X86;
- Windows X64;
- Linux X86;
- Linux X64;
- Raspberry Pi OS ArmV71;
- Linux Ubuntu Arm64.

МультиОберон это масштабируемая технология на основе систем ограничений с начальной точкой в виде синтаксиса Компонентного Паскаля. МультиОберон предназначен для работы из среды BlackBox и из командной строки.

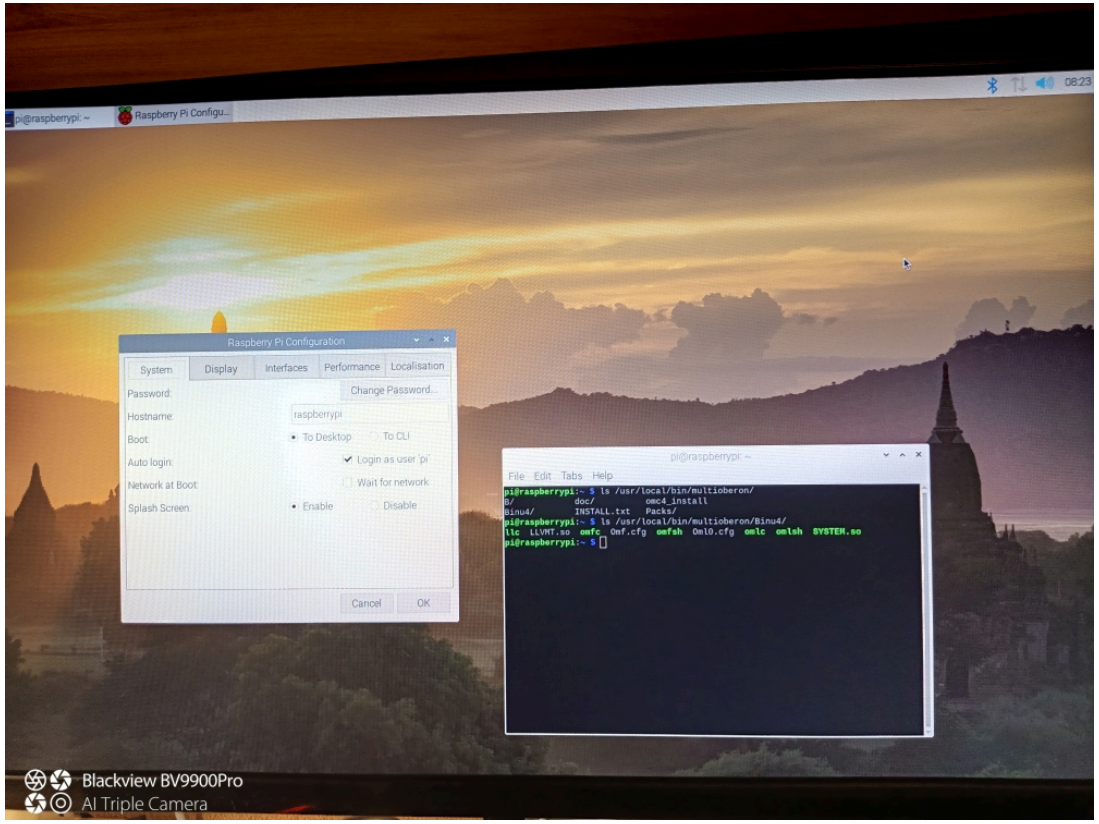
https://github.com/dvdagaev/Mob

Мини ПК Raspberry Pi 4 Model B 8Gb

В Raspberry Pi 4 используется однокристальная система Broadcom BCM2711. Кристалл включает в себя 4-ядерный 64-битный процессор Cortex-A72 с частотой 1,5 ГГц и графический процессор GPU VideoCore VI с частотой 500 МГц



Raspberry Pi OS для ARM32



pi@raspberrypi:~ $ uname -a

Linux raspberrypi 5.4.51-v7l+ #1333 SMP Mon Aug 10 16:51:40 BST 2020 armv7l GNU/Linux

Механизм переноса

1. Собираем компилятор в кодах C (Omf - OFront)
2. Переносим на ARM32 компилятор в кодах C
3. Собираем компилятор, корректируя коды C
4. Компиляция самого компилятора
5. Отладка линкера
6. Отладка загрузчика

1. Собираем библиотеку LLVM.so для ARM
2. Переносим на ARM компилятор в кодах LLVM
3. Собираем компилятор на ARM из кодов LLVM
4. Компиляция самого компилятора
5. Отладка линкера
6. Отладка загрузчика

Изменения в Kernel для ARM32

- Выделение пула динамической памяти:
  - Низкоуровневый вариант с предположением о начальном адресе adr := Api.mmap(01000000H, s, {}, {Api.MAP_PRIVATE, Api.MAP_ANONYMOUS}, -1, 0); (*x86*)
  - Система адресации отличается Api.posix_memalign(ptr, MIN_SZ, (s-1+MIN_SZ) DIV MIN_SZ * MIN_SZ) = 0 (*arm32*)
- Проблема когерентности кэша для APM – невозможность изменения кода после mprotect
  - Не работает для ARM, после mprotect нельзя грузить код Api.mprotect(adr, MIN_SZ, {Api.PROT_READ, Api.PROT_WRITE}) (*x86*)
  - Спасибо коллегам из stackoverflow - прояснили

Проблемы с линкером

- Генерация/вставка машинного кода для арифметических операции (gcc вставляет подобный код). Решение – использование объектных файлов с кодом.
- Явная компиляция llc из набора LLVM

```
pi@raspberrypi:~/mobdev/Mob $
nm Binu4/_arm_addsubdf3.o
0000000c T __adddf3
0000000c T __aeabi_dadd
00000000 T __aeabi_drsub
00000008 T __aeabi_dsub
00000304 T __aeabi_f2d
000002dc T __aeabi_i2d
00000360 T __aeabi_l2d
000002b8 T __aeabi_ui2d
0000034c T __aeabi_ul2d
00000304 T __extendsfdf2
00000360 T __floatdidf
000002dc T __floatsidf
0000034c T __floatundidf
000002b8 T __floatunsidf
00000008 T __subdf3
```

Другие типы релокаций в загрузчике объектных файлов ARM32

Существенно большее количество типов релокаций в объектных файлах. Из которых реально используются 5.

Другая архитектура и ассемблер.

Code	Name	Type	Class	Operation
2	R_ARM_ABS32	Static	Data	(S + A) | T
28	R_ARM_CALL	Static	Arm	((S + A) | T) - P
29	R_ARM_JUMP24	Static	Arm	((S + A) | T) - P
43	R_ARM_MOVW_ABS_NC	Static	Arm	(S + A) | T
44	R_ARM_MOVT_ABS	Static	Arm	S + A
138	…			

Дополнительный код для R_ARM_CALL, R_ARM_JUMP24

ARM инструкция имеет 32 бит. Для команд условного перехода диапазон относительных адресов ограничен +-1fffffc.

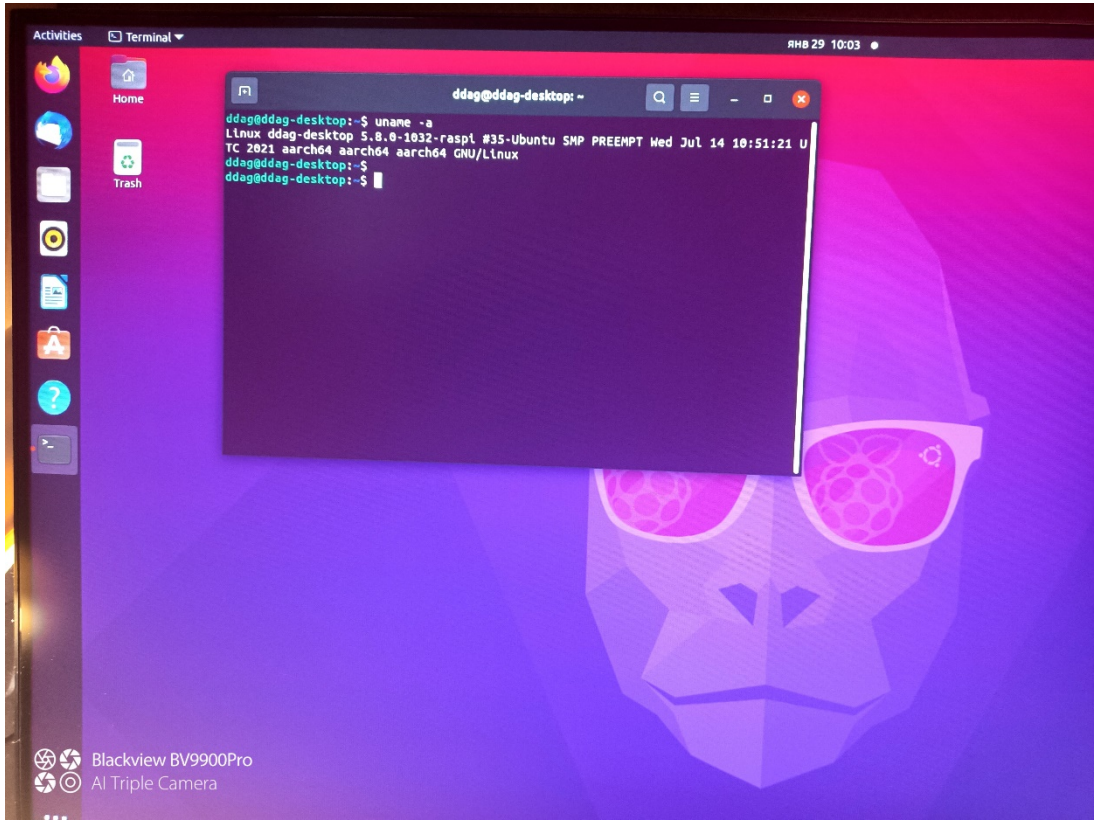
Относительный адрес зависит от места загрузки модуля.

Если относительный адрес выходит за рамки – вставляем машинный код

```
ldr r12,=DistantLabel
bx r12
```

R12 используется как scratch register

Ubuntu OS для ARM64 (AArch64)



ddag@ddag-desktop:~$ uname -a

Linux ddag-desktop 5.8.0-1032-raspi #35-Ubuntu SMP PREEMPT Wed Jul 14 10:51:21 UTC 2021 aarch64 aarch64 aarch64 GNU/Linux

Изменения в Kernel для AArch64

- Изменение размера адреса;
- Изменение размера блока;
- Изменение выравнивания указателя;
- Изменение типов целочисленных переменных адресной длины;

```
Block = POINTER TO RECORD
[untagged]
      tag: Type;
      last: @ADDR;
      (* arrays: last element *)
      actual: @ADDR;
      (* arrays: used during
mark phase *)
      first: @ADDR
      (* arrays: first element
*)
END;
```

Изменения в Api, Host

Разные заголовочные файлы системных библиотек

- _STAT_VER = 3 (* X86, ARM32 *) = 0 (* ARM64 *) = 1 (* X64 *)
- Api.__xstat(Api._STAT_VER, s, stt.st);

Изменение типов с 32 на 64 целочисленных полей структур, например, time_t

Другие типы релокаций в загрузчике объектных файлов Aarch64

Code	Name	Operation	Comment
257	R_AARCH64_ABS64	S + A	S + A
261	R_AARCH64_PREL32	S + A - P	S + A - P
311	R_AARCH64_ADR_GOT_PAGE	Page(G(GDAT(S+A))) -Page(P)	Page(G(GDAT(S+A))) -Page(P)
312	R_AARCH64_LD64_GOT_LO12_NC	G(GDAT(S+A))	G(GDAT(S+A))
275	R_AARCH64_ ADR_PREL_PG_HI21	Page(S+A) -Page(P)	Set an ADRP immediate value to bits [32:12] of the X; check that -232 ≤ X < 232
277	R_AARCH64_ ADD_ABS_LO12_NC	S + A	Set an ADD immediate value to bits [11:0] of X. No overflow check. Used with relocations ADR_PREL_PG_HI21 and ADR_PREL_PG_HI21_NC
278	R_AARCH64_ LDST8_ABS_LO12_NC	S + A	Set an LD/ST immediate value to bits [11:0] of X. No overflow check. Used with relocations ADR_PREL_PG_HI21 and ADR_PREL_PG_HI21_NC
285	R_AARCH64_ LDST32_ABS_LO12_NC	S + A	Set the LD/ST immediate value to bits [11:2] of X. No overflow check
286	R_AARCH64_ LDST64_ABS_LO12_NC	S + A	Set the LD/ST immediate value to bits [11:3] of X. No overflow check
282	R_AARCH64_JUMP26	S+A-P	Set a B immediate field to bits [27:2] of X; check that -227 ≤ X < 227
283	R_AARCH64_CALL26	S+A-P	Set a CALL immediate field to bits [27:2] of X; check that -227 ≤ X < 22

GOT – таблица глобальных офсетов в объектном файле

Активно реализуется идея позиционно-независимого кода. Но нужна таблица GOT.

В загрузчике Aarch64Relocator.Init реализует заполнение таблицы глобальных офсетов символами модулей.

Из документации:

- GOT is the address of the Global Offset Table, the table of code and data addresses to be resolved at dynamic link time. The GOT and each entry in it must be 64-bit aligned.
- GDAT(S+A) represents a 64-bit entry in the GOT for address S+A. The entry will be relocated at run time with relocation R_AARCH64_GLOB_DAT(S+A).
- G(expr) is the address of the GOT entry for the expression expr.

МультиОберон 1.2 для ARM32, ARM64

МультиОберон для ARM работает с различными бэкендами:

- Транслятором Ofront в язык C;
- Генератором кода LLVM.

МультиОберон для ARM это кроссплатформенный компилятор с поддержкой:

- Raspberry Pi OS ArmV71;
- Linux Ubuntu Arm64.

МультиОберон использует систему ограничений с начальной точкой в виде синтаксиса Компонентного Паскаля.

МультиОберон для ARM предназначен для работы из командной строки.

МультиОберон для ARM поддерживает режимы оптимизирующей компиляции для обоих бэкендов.

МультиОберон для ARM реализует динамическую загрузку модулей в виде объектных файлов.

Что не поддерживает МультиОберон

МультиОберон не поддерживает полный механизм ABI динамического вызова функций, как в BlackBox. Поэтому в Kernel данные механизмы не вставлены, равно как и мета-интформация относящаяся к ним. Естественно, вызов процедуры по имени присутствует.