Методические указания к выполнению лабораторной работы "Комплект средств разработки программного обеспечения процессора Байкал-Т1"

Оглавление

Лабораторная работа №1. Комплект средств разработки программного обеспечения процессора Байкал-Т1

Цель работы: изучение состава и функциональных возможностей комплекта средств разработки программного обеспечения (SDK), изучение кросс-компиляции под Байкал-Т1, ознакомление с работой загрузчика u-boot.

Введение

Baikal-T1 – отечественная система на кристалле на базе архитектуры нового поколения MIPS Warrior P-class P5600. Это современный энергоэффективный процессор с большим набором высокоскоростных интерфейсов, предназначенный для широкого диапазона целевых устройств потребительского и B2B сегментов.

Процессор Байкал-Т1 производится с использованием 28-нанометровой технологии, его энергопотребление не превышает 5 Вт.

Блок-схема процессора Байкал-Т1 представлена на рисунке 1. Процессор включает в себя следующие логические блоки:

- Двухъядерный микропроцессорный кластер;
- Контроллер памяти;
- Высокоскоростные интерфейсы ввода-вывода;
- Низкоскоростные интерфейсы для подключения периферийных устройств;
- Подсистема отладки;
- Высокоскоростная внутрипроцессорная шина.

Логические блоки системы на кристалле взаимодействуют между собой с помощью высокоскоростной внутрипроцессорной шины. Она организована по принципу матричного переключателя и обеспечивает соединение входов с выходами по схеме «многие со

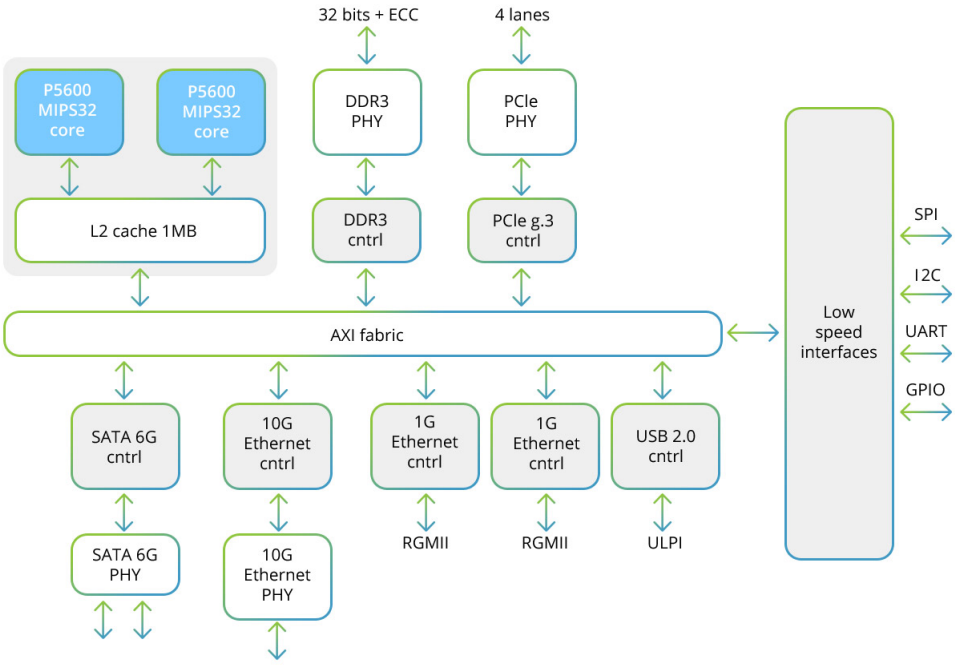

Рис. 1 Блок-схема Байкалl-T1

многими». В глобальном адресном пространстве процессора Байкал-T1 логические блоки идентифицируются своими адресами.

Особенностями процессора Байкал-T1 являются:

- Два когерентных микропроцессорных ядра серии P5600 с архитектурой MIPS32®;
- Рабочая частота ядра серийных изделий – до 1,2 ГГц;
- Кэш данных и кэш инструкций уровня L1 размером 64 Кб;
- Контроллер управления когерентностью со встроенным восьмиканальным ассоциативным кэшем L2 размером 1 Мб;
- Встроенная память 32 Кб для сбора данных от трассировщика PDTrace;
- Выделенный порт EJTAG, поддерживающий отладку кода, исполняющегося на нескольких процессорах.

Комплект средств разработки программного обеспечения для микропроцессора Байкал-T1 (далее SDK) содержит кросс-компилятор языков C и C++, редактор связей, отладчик, утилиты и системные библиотеки, достаточные для разработки системного и прикладного ПО, исполняемого процессором Байкал-T1. В качестве платформы разработчика используется ПК архитектуры Intel под управлением ОС Linux. Скомпилированное

ПО может исполняться как под эмулятором на платформе разработчика, так и на физической плате с процессором Байкал-Т1.

В составе пакета SDK поставляются средства для компиляции, сборки и отладки программных продуктов для платформы Байкал-Т1:

- Образ ядра ОС Linux, содержащий исходный код ОС и код целевой платформы, драйверы для всех реализованных устройств и конфигурационный файл для сборки. Поставляемая версия ядра – 4.4.24 или более новая;
- Образ корневой файловой системы (InitRD) в виде сжатого диска размером 32 Мб. Файловая система включает в себя минимальный набор необходимых утилит и основные библиотеки;
- Набор драйверов для периферийных устройств, контроллеры которых входят в состав микросхемы Байкал-Т1, в исходных кодах и в скомпилированном виде;
- Средства для кросс-компиляции на основе комплекса программ gcc из-под x86 для целевой архитектуры MIPS32®, в том числе отладчик gdb;
- Функциональный эмулятор процессора Байкал-Т1 на основе ПО с открытым кодом QEMU.

  Пользовательский эмулятор позволяет выполнять приложение, скомпилированное при помощи средств кросс-компиляции в файл формата elf под архитектуру MIPS32.

Структура директорий SDK приведена на рисунке 2.


```
baikal
├── bin – бинарные исполняемые файлы (симулятор)
├── doc – документация
├── img – образы операционной системы и файловых систем
├── lib – библиотеки для работы с адаптером EJTAG
├── src
│   ├── bios – исходные коды и пакет сборки прошивки ROM (BIOS) платы БФК
│   ├── boot – исходный код начального загрузчика платформы Baikal-T1
│   ├── configs – конфигурационные файл для сборки компонентов пакета
│   ├── dfu-util – исходные коды пакета dfu-utils
│   ├── examples – файлы примеров кода
│   ├── initrd – исходные коды и каркас корневой файловой системы
│   ├── kernel – исходные коды ядра ОС Linux
│   ├── openocd – исходные коды пакета OpenOCD
│   ├── qemu – исходные коды симулятора QEMU
│   ├── u-boot – исходные коды начального загрузчика U-Boot
│   └── x-tools – исходные коды средств кросс-компиляции и утилит
└── usr
    ├── eclipse – графическая среда разработки ПО
    ├── scripts – вспомогательные скрипты для модификации и запуска образов системы
    └── x-tools – предкомпилированные средства кросс-компиляции.
```

Рис. 2 Структура директорий

Задание 1. Установка SDK

SDK распространяется в виде самораспаковывающегося архива, файла с именем вида `sdk_baikal_mips_xxx.run`. Необходимо скопировать в домашнюю директорию установчный скрипт из папки `/srv/sdk/sdk_baikal_mips_xxx.run`.

Для его установки на ПК под управлением ОС Linux необходимо произвести описанные ниже действия.

1. Установить атрибут, разрешающий исполнение файла:

   ```
   $ chmod +x sdk_baikal_mips_xxx.run
   ```

2. Запустить программу:

   ```
   $ ./sdk_baikal_mips_xxx.run
   ```

   Программа установится в подкаталог baikal, который она создаст в текущем каталоге. Результат установки SDK представлен на рисунке 3

Для проверки корректности установки необходимо произвести запуск симулятора с поставляемым образом операционной системы и пустым файлом жесткого диска.

1. Перейти в директорию с управляющими скриптами:

```
Creating directory baikal
Verifying archive integrity... All good.
Uncompressing BAIKAL-T SoC BSP Pack. Version 4.08 build 84 (Fri Nov 11 14:50:17 MSK 2016)  100%


                BAIKAL ELECTRONICS BAIKAL-T1 BFK BOARD BSP PACK
                -----------------------------------------------
    This package contains Board Support Package for Baikal Electronics BFK board
including Software Development Kit (SDK) for Baikal-T series SoCs.
It provide our customers with a comprehensive starting point for their Linux
development efforts on MIPS32 platform. These packages are developed and tested
to support 34K, 74K and P5600 MIPS processors. This SDK is tested for use
with a given processor and its supporting development system, ensuring an
operational toolchain, kernel and specific peripherals that are ready to use
together within a fixed configuration for specific hardware reference platforms.

    Baikal Linux SDK typically include Linux kernel, device drivers, libraries,
simulator, and GNU Tools like compilers, linkers, etc. The documentation will
provide detailed information on the version of the kernel, glibc, gcc, etc.,
as well as information about simulator is included within a specific SDK.

    For more information about BSP and Quick Start Guide please read README file.
    For detailed information please read document "Software Development Kit for
Baikal-T platform".
```

Рис. 3 Результат установки SDK

```
$ cd baikal/usr/scripts
```

2. Запустить симулятор с образом операционной системы и жесткого диска по умолчанию:

   ```
   $ ./run-qemu-mipsel.sh -test
   ```

3. Дождаться загрузки операционной системы Linux. После запуска необходимых сервисов, интерпретатор командной строки выдаст сообщение "Please press Enter to activate this console". Нажмите клавишу `<Enter>`. Из командной строки консоли выполните команду вывода версии ядра:

   ```
   # uname -a
   ```

4. Проверьте версию ядра, она должна быть не ниже 4.4.24

5. После проверки наберите команду `halt`, и симуляция остановится. Для выхода из симулятора нажмите сочетание клавиш `<CTRL+a> <x>` в окне терминала.

Комплект программных средств разработки для платформы Baikal-T1 содержит готовый образ операционной системы Linux. Это позволяет производить разработку прикладного программного обеспечения без необходимости сборки ядра операционной системы и корневой файловой системы. Однако возможно, что в процессе разработки возникнет необходимость подключения дополнительных модулей или опций ядра ОС Linux а также модификации корневой файловой системы. В данной ситуации возникает необходимость сборки модифицированного загрузочного образа операционной системы.

Образ операционной системы включает в себя 3 компонента: загрузчик, ядро ОС Linux и образ корневой файловой системы. Исходные коды компонентов расположены в директории `<PATH_TO_SDK>/baikal/src`.

Скрипты, входящие в состав SDK и установленные в каталоге `<PATH_TO_SDK>/baikal/usr/scripts` позволяют пересобирать указанные выше образы.

Скрипт `build-boot-img.sh` осуществляет полную сборку ядра Linux, образа корневой файловой системы и загрузчика.

Формат запуска скрипта:

```
./build-boot-img.sh qemu -q
```

Параметр `qemu` указывает, что сборка производится под эмулятор QEMU, входящий в состав SDK. Полный список поддерживаемых значений параметра и опций для установленной версии скрипта можно получить командой `./build-boot-img.sh -h`.

Результат окончания сборки представлен на рисунке 4.

```
Flash image build utility. (c) 2015,2016 Baikal Electronics JSC.
Version 1.06 (Wed Apr 13 12:16:15 MSK 2016)

+-------------+-------------+-------------+-------------+-------------+--------+
| Flash Block | Start Offset |  Max Length  | Flash Offset | Zero Padding | Status |
+-------------+-------------+-------------+-------------+-------------+--------+
| BOOTLOADER  |  0x00000000  |  0x00080000  |  0x0007fd00  |  0x00000300  | OK     |
| ENVSET      |  0x00080000  |  0x00020000  |  0x00090000  |  0x00010000  | OK     |
| FIRMWARE    |  0x000a0000  |  0x00020000  |  0x000a0197  |  0x0001fe69  | OK     |
| FDT         |  0x000c0000  |  0x00040000  |  0x000c390d  |  0x0003c6f3  | OK     |
| MULTIIMAGE  |  0x00100000  |  0x00b00000  |  0x00bd00f7  |  0x0002ff09  | OK     |
+-------------+-------------+-------------+-------------+-------------+--------+

  Created flash image file 'bfk-mips.rom' (12288 kbytes)

  INFO:  Build process is done

  #####  #     #  #####   #####  #######  #####   #####
 #     # #     # #     # #     # #       #     # #     #
 #       #     # #       #       #       #       #
  #####  #     # #       #       #####    #####   #####
       # #     # #       #       #            #       #
 #     # #     # #     # #     # #       #     # #     #
  #####   #####   #####   #####  #######  #####   #####
```

Рис. 4 Результат успешной сборки SDK

Файлы, полученные в результате успешной сборки, копируются в каталог `<PATH_TO_SDK>/baikal/img`.

Для упрощения запуска образа ОС Linux на симуляторе пакет SDK содержит скрипты в директории `<PATH_TO_SDK>/baikal/usr/scripts`. Скрипты позволяют производить запуск симулятора с подключенным или отключенным образом жесткого диска.

Пример строки запуска образа ОС приведен ниже:

```
./<PATH_TO_SDK>/baikal/usr/scripts/run-qemu-mipsel.sh -img
<PATH_TO_SDK>/baikal/img/linux-baikal-mipsel.elf
```

Результат запуска образа ОС представлен на рисунке 5.

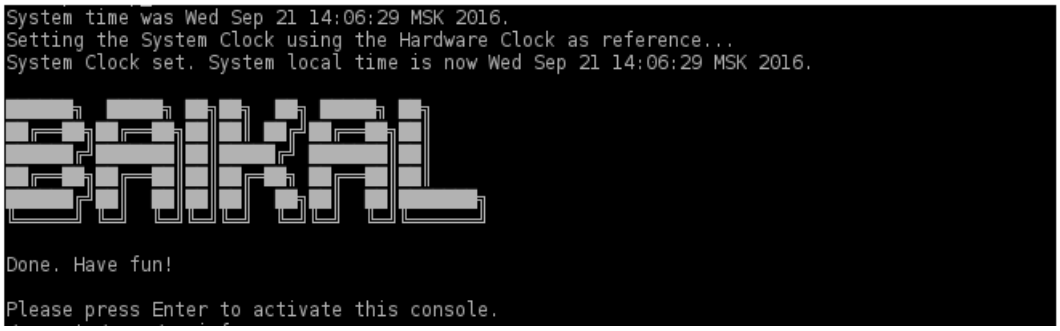

Рис. 5 Результат запуска образа ОС

Задание 2. Изучение кросс-компиляции под Байкал-Т1

Результатом кросс-компиляции является исполняемый код для платформы, отличной от той, на которой выполняется сама компиляция. Такой инструмент полезен, когда требуется получить код для платформы, экземпляров которой нет в наличии, или в случаях когда компиляция на целевой платформе невозможна или нецелесообразна (например, это касается мобильных систем или микроконтроллеров с минимальным объёмом памяти).

Рассмотрим применение кросс-компиляции на примере добавления собственного файла в корневую файловую систему загружаемого образа ОС Linux. Для это необходимо выполнить указанный список действий.

1. Открыть с помощью текстового редактора скрипт по следующему адресу

   `<PATH_TO_SDK>/baikal/usr/scripts/build-initrd-img.sh`, в нем необходимо добавить желаемый пакет в переменнную `EXTRA_PKG`:

```
# Include kernel modules:
KERNELMODULES=

# Include packages ('busybox' is always included):
PACKAGES=   # <--- list of packages
I2CTOOLS=1  # <--- default packages
LMSENSORS=1 # <--- default packages
ETHTOOL=1   # <--- default packages
DROPBEAR=
KEXEC=
FSCHECK=
FBTEST=
BRDINFO=
# Additional (user defined) packages can be added as a list of packagecd src/
# names to the EXTRA_PKG variable, for example:
# EXTRA_PKG="gpiotest xutils xxxtools"
```

```
# You need to add package source files to
# {BSP_ROOT}/sdk_bt/baikal/src/initrd/programs/ directory
# and create appropriate makefiles for them. For example,
# if you need packages 'gpiotest', 'xutils' and 'xxxtools',
# add the following to {BSP_ROOT}/sdk_bt/baikal/src/initrd/programs:
# Directory "gpiotest" and makefile "gpiotest.mk",
# Directory "xutils" and makefile "xutils.mk",
# Directory "xxxtools" and makefile "xxxtools.mk",
EXTRA_PKG="crossexample"
```

2. На сервере по адресу `/srv/sdk/crossexample` находится директория crossexample с указанными ниже файлами.

3. Разместить директорию crossexample по следующему адресу:

   `<PATH_TO_SDK>/baikal/src/initrd/programs/`

4. Добавить в директорию crossexample файл `crossexample.c`:

```
#include <stdio.h>
int main (int argc, char * argv[])
{
        printf("\n\rHello World\n\r");
}
```

5. Добавить в директорию crossexample следующий `Makefile`:

```
GCC5 := $(shell echo `gcc -dumpversion
| cut -f1-2 -d.` \>= 5.1 | sed -e 's/\./*100+/g' | bc )
CC            := $(CROSS_COMPILE)gcc
LD            := $(CROSS_COMPILE)ld
STRIP         := $(CROSS_COMPILE)strip
TARGETS        = crossexample
CFLAGS         = -Wall -O2
CFLAGS        += -mno-mips16 -mabi=32 -mno-micromips
ifeq ($(GCC5),1)
CPPFLAGS+= -mfp64 -Wa,-mmsa -mips32r5 -mtune=p5600
endif
LDFLAGS        =
SDK            = $(shell cat VERSION 2>/dev/null)
ifneq ($(SDK),)
CFLAGS        += -DSDK=\"$(SDK)\"
```

```
endif
all: Makefile $(TARGETS)
crossexample: crossexample.o
        $(CC) $(CFLAGS) $< -o $@
c.o:
        $(CC) $(CFLAGS) $< -o $@
clean:
        rm -f *.o $(TARGETS)
install: all
        mkdir -p $(DESTDIR)
        install --mode=755 --strip --strip-program=
$(STRIP) --target-directory=$(DESTDIR)/sbin $(TARGETS)
.PHONY:
        all clean install $(TARGETS)
```

6. Добавить makefile `crossexample.mk` по следующему адресу:

   `<PATH_TO_SDK>/baikal/src/initrd/programs`

```
include Makefile.env
phonys += crossexample

all:    crossexample
        @echo "Build completed for "$^

crossexample:
        $(MAKE) -C $@ CC="$(CC)"  AR=$(AR) DESTDIR=$(PREFIX)

install:
        @mkdir -p $(PREFIX)
        $(MAKE) -C crossexample CC="$(CC)" DESTDIR=$(PREFIX) install

clean:
        cd crossexample && $(MAKE) clean

.PHONY: $(phonys)
```

7. Заключительным шагом на пути к добавлению собственной программы в `initrd` является пересборка с помощью команды `./build-boot-img.sh qemu -a`. Добавленный пакет будет находиться в директории `sbin`.

Сдедующий этап знакомства с кросскомпиляцией - создание bare-metal теста интерфейса uart его запуск на QEMU. На данном этапе может возникунть вопрос, почему начальная программа это тест интерфейса, а не классический hello world. В случае bare-metal программ для вывода информации в консоль необходимо иметь работающий интерфейс, в нашем случае uart, таким образом, данная задача и есть hello world, только с учетом того, что данный тест запускается на неинициализированном железе.

Для это следует выполнить ряд нижеописанных действий.

1. Требуется перейти в папку `<PATH_TO_SDK>/baikal/src/examples/bare_metal_uart_qemu`. Если данная директория отсутствует, то ее необходимо скопировать из папки `/srv/sdk/` на сервере.

   Состав директории `bare_metal_uart_qemu`:

   - `build` - бинарные файлы;

     В составе данной директории находятся файлы `uart.bin` - бинарный файл, `uartdis.text` - файл на ассемблере, `uart` - исполняемый файл.
   - `include` - заголовочные файлы;
   - `obj` - объектные файлы;
   - `source` - исходные файлы.

2. Необходимо ознаокмиться с файлом `test.lds`, он используется линковщиком и осдержит информацию о том, по каким адресам будет слинкован исполняемый файл. Для запуска на эмуляторе адрес должен быть 0x9FC00000, что соответсвует spi-flash карты памяти процессора Байкал-Т1. Если файл слинковать по другим адресам, то тест не будет работать.

3. Воспользоваться утилитой `make` для создания необходимых файлов.

4. В случае возникновения ошибок необходимо внести изменение в путях до средств кросс-компиляции в makefile:

```
. . .
ROOT          ?= $(shell readlink -f ../../../)
CROSS_COMPILE ?=
$(ROOT)/usr/x-tools/mipsel-unknown-linux-gnu/bin/mipsel-unknown-linux-
. . .
```

Или с помощью `export` внести собственное значение `CROSS_COMPILE` через консоль.

Команда для запуска QEMU из директории `hello_world/build`:

```
./<PATH_TO_SDK>/baikal/bin/qemu-system-mipsel -bios helloworld.bin -D trace -d asm
-singlestep -machine baikal-t -net none -icount 0 -vnc none -serial stdio
```

Зафиксируйте полученный результат.

Задание 3. Запуск bare-metal теста с помощью загрузчика u-boot

U-Boot – это кроссплатформенный загрузчик для встраиваемых систем, используемый в качестве загрузчика по умолчанию. Он может работать на множестве поддерживаемых архитектур, включая PPC, ARM, MIPS, x86, m68k, NOIS и Microblaze.

U-boot позволяет изменить переменные окружения, отвечающие за настройки сети, в его собственной консоли, но не имеет возможности их сохранить. Таким образом, если планируется многократная загрузка тестов, то наиболее удачной конфигурацией будет перевод хост-машины и загрузчика в отдельную подсеть, так в случае непредвиденных ситуаций не будет оказано влияния на существующие у пользователя сети. Необходимо убедиться, что хост-машина и загрузчик находятся в одной подсети, если это не так, то произведите соответствующую настройку согласно описанию в приложении.

Тест находится в директории по следующему адресу:

`<PATH_TO_SDK>/baikal/src/examples/bare_metal_uart_qemu`

Необходимо убедиться, что в файле `test.lds` линковка производится на начало оперативной памяти (адрес `0xA1000000`) и пересобрать тест.

Требуется скопировать бинарный файл `uart.bin` из папки `build` в папку

`/srv/tftp/<YOUR_CARD>`. После этого нужно с помощью комнады `ls -l` проверить, что права доступа соответсвуют 755.

После выполнения данных операций представляется возможным начать работу с платой. Для этого необходимо на стадии выбора конфигурации загрузки в меню u-boot выбрать консоль, пример представлен на рисунке 6. Для перехода в меню загрузки необходимо сразу после включения питания платы нажать клавишу `<S>` (только для платы miniATX).

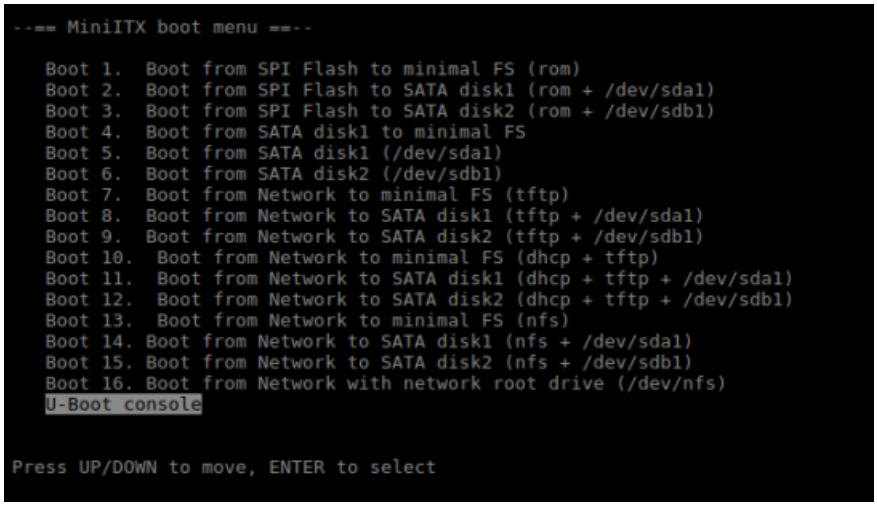

Рис. 6 Выбор консоли u-boot

После входа в консоль u-boot следует ознакомиться с командой, позволяющей загру-

зить тест в оперативную память. Вывод команды `help` приведен ниже:

```
| BAIKAL # help tftpboot
| tftpboot - boot image via network using TFTP protocol
| Usage:
| tftpboot [loadAddress] [[hostIPaddr:]bootfilename]
| BAIKAL #
```

Особенностью архитектуры MIPS является включение MMU сразу после подачи питания, поэтому тест необходимо собирать и загружать по виртуальным адресам. Адрес оперативной памяти `0xA1000000`. Поэтому если тест с названием бинарного файла `uart.bin`, собранный на начало оперативной памяти, и сетевые настройки соответствуют вышеописанным, то команда загрузки теста с tftp-сервера будет выглядеть следующим образом:

`tftpboot 0xA100000 <hostIPaddr>:uart.bin`, где вместо `<hostIPaddr>` необходимо вставить ip-адрес хост-машины.

После чего u-boot сообщит об удачной операции загрузки:

```
| BAIKAL # tftpboot 0xa1000000 192.168.0.20:uart.bin
| Speed: 100, full duplex
| Using dwmac.bf05e000 device
| TFTP from server 192.168.0.20; our IP address is 192.168.0.25
| Filename 'uart.bin'.
| Load address: 0xa1000000
| Loading: #
|   465.8 KiB/s
| done
| Bytes transferred = 15744 (3d80 hex)
```

Также следует помнить,что запускать тест необходимо с адреса стартовой функции, а не с начала оперативной памяти. Адрес стартовой функции можно найти в файле `uartdis.text`, расположенном в папке `build`.

Далее остается только запустить загруженный тест с помощью нижеописанной команды `go`:

```
| BAIKAL # help go
| go - start application at address 'addr'
| Usage:
| go addr [arg ...]
|     - start application at address 'addr'
|       passing 'arg' as arguments
| BAIKAL #
```

Формат запуска команды `go`: `go 0xa1000000`

Важно то, что выполнять основную инициализацию процессора в тестах уже не требуется, поскольку это делает u-boot.

В файлах `boot.S`, `boot_misc.S` директории `source` описана инициалицация процессора и периферии, рекомендуется самостоятельно ознакомиться с их содержимым.

Задание 4. Загрузка ОС Linux с помощью загрузчика u-boot

Для выполнения данного задания, так же как и предыдущего, необходим настроеный tftp-сервер.

Для выполнения загрузки Linux с помощью загрузчика u-boot необходимо выполнить следующие пункты.

1. Необходимо пересобрать ядро и корневую файловую систему с помощью скрипта:

   `./build-boot-img.sh bfk2 -a`

2. Необходимо скопировать файлы: `vmlinux.bin`, `initrd.gz`, `baikal.dtb` из папки `img` в папку `/srv/tftp/<YOUR_CARD>`. После этого нужно проверить права доступа аналогично тому, как это делалось в задании 3:

3. Нужно перезагрузить плату и в меню загрузчика выбрать пункт 10 (dhcp+tftp), пример представлен на рисунке 7. В меню загрузчика можно попасть нажатием клавиши `<S>` сразу после включения питания платы (только для плат miniATX).

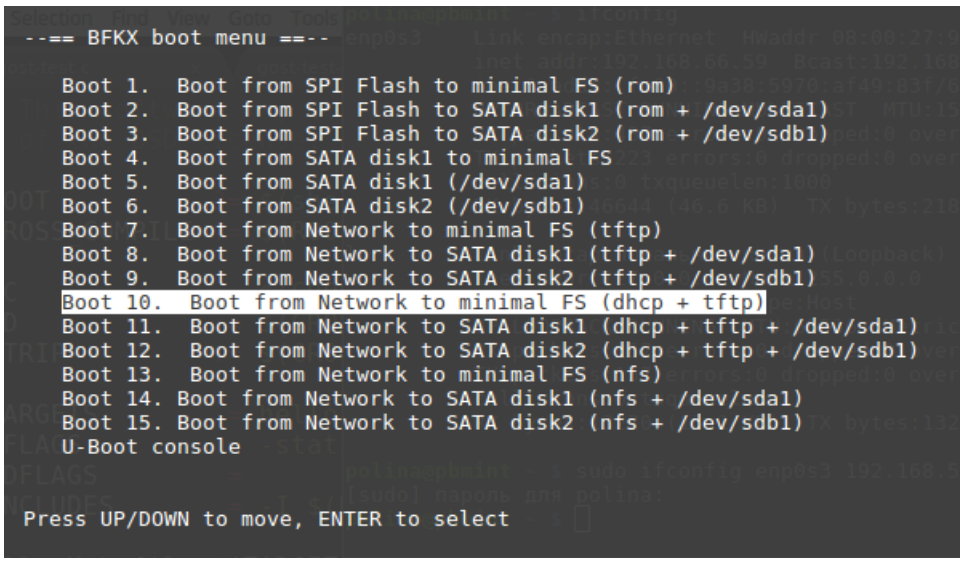

Рис. 7 Выбор пункта в меню загрузчика

После загрузки системы следует проверить наличие файла `crossexample` в `sbin` и убедиться, что он выполняется.

Задание 5. Загрузка приложения в ОС Linux через tftp

Для выполнения данного задания требуется применить все навыки, полученные в ходе предыдущих четырех заданий данной лабораторной работы. Вам предлагается самостоятельно выполнить все пункты данного задания.

С помощью кросс-компилятора из SDK необходимо самостоятельно получить исполняемый файл программы hello world. После чего запустить ОС с помощью u-boot. Используя встроенный tftp-клиент (информацию можно получить с помощью команды `help`), необходимо загрузить полученный исполняемый файл в директорию tmp и выполнить его.

В случае возникновения затруднений следует обратиться к указаниям по выполнению задания 5.

Указания по выполнению задания 5

Проверить наличие исполняемого файла в папке /srv/tftp/<YOUR_CARD> на хост-машине.

Выполнить команду `tftp -g -r <filename> <serverip>` из директории `tmp`.

С помощью команды `chmod +x <file name>` сделать ваш файл исполняемым.

Контрольные вопросы

1. На базе какой архитектуры построен процессор Байкал-T1? Каковы ее преимущества и недостатки по сравнению с ARM и x86?
2. Что входит в состав комплекта средств разработки программного обеспечения для Байкал-T1?
3. Для чего предназначен пользовательский эмулятор QEMU?
4. Можно ли осуществлять неполную пересборку ОС после добавления собственной программы в `initrd`? Пересборку каких частей осуществлять обязательно?
5. Что такое кросскомпиляция и для чего она предназначена?
6. Для чего нужен загрузчик u-boot и какие функции он обеспечивает?

Приложение

Для успешного соединения между u-boot и tftp-сервером, запущенном на хост-машине, необходимо, чтобы они находились в одной подсети.

Настройка адресов в u-boot

Для изменения адресов в u-boot необходимо воспользоваться командой `setenv`:

```
setenv serverip <ip address>
setenv ipaddr <ip address>
```

Для просмотра параметров используется команда `printenv`.

`ipaddr` отвечает за адрес хоста в сети, а `serverip` отвечает за адрес, по которому пойдет обращение к серверу tftp, то есть в этом поле необходимо указать адрес хост-машины, на которой будет запущен tftp-сервер (также важно, чтобы сервер, работающий на хост-машине, слушал все адреса). Необходимо также произвести соответствующую конфигурацию сетевого интерфейса хост-машины, её адрес должен быть точно таким же, как и адрес, заданный в поле `serverip`.

