Gestor de arranque - U-Boot

Objetivos: Compilar e instalar el gestor de arranque U-Boot. Uso básico de los comandos de U-Boot.

Dado que el gestor de arranque es la primer pieza de Software ejecutada por una plataforma Hardware, el procedimiento de instalación del gestor de arranque es específico para dicha plataforma.

En general existen dos casos:

- El procesador no ofrece ninguna facilidad para la instalación del gestor de arranque, en cuyo caso se debe utilizar JTAG para inicializar el almacenamiento flash y escribir el código del gestor de arranque en la misma. Es necesario un conocimiento detallado del Hardware para poder realizar estas operaciones.
- El procesador ofrece un monitor, implementado en ROM, a través del cual es más simple acceder a la memoria.

Instalación de los paquetes necesarios

Instale los paquetes necesarios para este laboratorio:

```
sudo apt-get install bc
```

Configuración de la comunicación serie con la placa

Para comunicarse con la placa a través del puerto serie, instale el programa de comunicaciones `picocom`:

```
sudo apt-get install picocom
```

Es necesario que pertenezca al grupo `dialout` para que tenga permisos para escribir sobre la consola serie:

```
sudo adduser $USER dialout
```

Puede consultar el nombre de usuario con el comando:

```
whoami
```

Nota: Debe salir del sistema y volver a entrar para que los cambios de grupo se hagan efectivos.

Ejecute `picocom -b 115200 /dev/pty/2`, para iniciar una comunicación serie en `/dev/pty/2`, con un baudrate de 115200. Si desea salir de `picocom`, presione `[Ctrl][a]` seguido de `[Ctrl][x]`.

Configuración de U-Boot

Vaya al directorio de los laboratorios `${PROJECT_ROOT}/bootloader`.

Descargue U-Boot:


```
git clone git://git.denx.de/u-boot.git
cd u-boot
git checkout v2019.04
```

Recomendamos entienda los pasos de configuración y compilación, leyendo el archivo `README`, y especialmente la sección *Building the Software*.

Básicamente, necesitaremos:

- Asignar a la variable de entorno `CROSS_COMPILE` el valor `arm-linux-` que es el alias que definimos en el primer laboratorio o `arm-unknown-linux-gnueabihf-` que seria el nombre completo (esto ya lo realiza el script de inicializacion `dev_qemu_glibc`
- Verificar que la variable de entorno `PATH` contiene la ruta al juego de herramientas generado en el laboratorio anterior `${PROJECT_ROOT}/tools/arm-unknown-linux-gnueabihf/bin` (esto ya lo realiza el script de inicializacion `dev_qemu_glibc`
- Ejecutar `make <nombre>_defconfig`, donde `<nombre>` es el nombre de la placa que vamos a utilizar. En nuestro caso es `vexpress_ca9x4_defconfig`
- Finalmente, ejecutamos `make`, para construir U-Boot

Nota: es posible acelerar la compilación usando la opción -jX de make, donde X es el número de trabajos en paralelo usados para compilar. En general un buen valor es duplicar el número de nucleos de CPU.

Una vez finalizado el proceso de compilación, se genera el archivo de U-Boot `u-boot`.

Probando U-Boot

Para probar la imagen de u-boot ejecutamos el siguiente comando:

```
qemu-system-arm -M vexpress-a9 -m 128M -nographic -kernel u-boot
```

Deberiamos ver una salida similar a la siguiente:

```
U-Boot 2018.09 (Sep 15 2018 - 13:37:39 +0000)

DRAM:  128 MiB
WARNING: Caches not enabled
Flash: 128 MiB
MMC:   MMC: 0
*** Warning - bad CRC, using default environment

In:    serial
Out:   serial
Err:   serial
Net:   smc911x-0
Hit any key to stop autoboot:  0
```

Para salir de qemu presionar `Ctrl-a` y luego `x`.

Si muestra un error como el siguiente:

```
pulseaudio: set_sink_input_volume() failed
pulseaudio: Reason: Invalid argument
pulseaudio: set_sink_input_mute() failed
pulseaudio: Reason: Invalid argument
```


Hay que ejecutar qemu asignando la variable de entorno `QEMU_AUDIO_DRV` en `none`:

```
QEMU_AUDIO_DRV=none qemu-system-arm -M vexpress-a9 -m 128M -nographic \
  -kernel u-boot
```

Interrumpa la cuenta regresiva para entrar a la línea de comandos de U-Boot:

```
=>
```

Podemos ver que al no estar asignada o disponible la zona de variables de entorno del usuario, utiliza el entorno por defecto:

```
*** Warning - bad CRC, using default environment
```

Para probar la comunicacion serie entre el dispositivo (emulado por qemu) y una terminal serie como ser `picocom` (que seria el escenario real) podemos agregar una salida serie en qemu a traves del argumento `-serial pty`. Por lo tanto, podemos ejecutar qemu de la siguiente manera:

```
QEMU_AUDIO_DRV=none qemu-system-arm -M vexpress-a9 -m 128M -nographic \
  -kernel u-boot -serial pty
```

Esto va a levantar la maquina virtual, pero en este caso, qemu entra en modo `monitor`:

```
QEMU 2.11.1 monitor - type 'help' for more information
(qemu) qemu-system-arm: -serial pty: char device redirected to /dev/pts/1 (label serial0)
```

donde `/dev/pts/1` puede contener otro numero de puerto. Ahora podemos conectarnos a traves de picocom:

```
picocom -b 115200 /dev/pty/1
```

Dado que la maquina virtual ya levanto, podemos reiniciarla desde el monitor de `qemu` con el comando `system_reset`.

Ahora volviendo a la terminal serie, podemos ver el inicio de u-boot.

En U-Boot, escriba el comando `help`, y explore alguno de los comandos disponibles.

También es posible consultar la versión de U-Boot con el comando `version`.

Listado comandos U-Boot

```
?       - alias for 'help'
base    - print or set address offset
bdinfo  - print Board Info structure
bootefi - Boots an EFI payload from memory
bootelf - Boot from an ELF image in memory
bootm   - boot application image from memory
bootp   - boot image via network using BOOTP/TFTP protocol
bootvx  - Boot vxWorks from an ELF image
bootz   - boot Linux zImage image from memory
cmp     - memory compare
cp      - memory copy
crc32   - checksum calculation
dhcp    - boot image via network using DHCP/TFTP protocol
echo    - echo args to console
env     - environment handling commands
erase   - erase FLASH memory
```



```
exit    - exit script
ext2load- load binary file from a Ext2 filesystem
ext2ls  - list files in a directory (default /)
ext4load- load binary file from a Ext4 filesystem
ext4ls  - list files in a directory (default /)
ext4size- determine a file's size
false   - do nothing, unsuccessfully
fatinfo - print information about filesystem
fatload - load binary file from a dos filesystem
fatls   - list files in a directory (default /)
fatsize - determine a file's size
fdt     - flattened device tree utility commands
flinfo  - print FLASH memory information
fstype  - Look up a filesystem type
go      - start application at address 'addr'
help    - print command description/usage
iminfo  - print header information for application image
load    - load binary file from a filesystem
loop    - infinite loop on address range
ls      - list files in a directory (default /)
md      - memory display
mii     - MII utility commands
mm      - memory modify (auto-incrementing address)
mmc     - MMC sub system
mmcinfo - display MMC info
mw      - memory write (fill)
nm      - memory modify (constant address)
part    - disk partition related commands
ping    - send ICMP ECHO_REQUEST to network host
printenv- print environment variables
protect - enable or disable FLASH write protection
pxe     - commands to get and boot from pxe files
reset   - Perform RESET of the CPU
run     - run commands in an environment variable
save    - save file to a filesystem
saveenv - save environment variables to persistent storage
setenv  - set environment variables
showvar - print local hushshell variables
size    - determine a file's size
source  - run script from memory
sysboot - command to get and boot from syslinux files
test    - minimal test like /bin/sh
tftpboot- boot image via network using TFTP protocol
true    - do nothing, successfully
version - print monitor, compiler and linker version
```

Soporte para flash

Para poder probar el soporte para flash bajo `QEMU` es necesario disponer de una versión mayor o igual a la 2.9 (verifique la salida del comando `qemu-system-arm --version`). Versiones anteriores contienen un bug en el soporte para flash CFI (Common Flash Interface) que hacen

inutilizable dicha funcionalidad en la maquina vexpress-a9.

En primer lugar debemos crear dos archivos que van ser el contenido de nuestra memoria flash. La maquina Versatile Express posee dos bancos de memoria flash de 64Mb cada uno.

Para crear dichos bancos, estando en el directorio `${PROJECT_ROOT}`, ejecutamos lo siguiente:

```
dd if=/dev/zero of=./bootloader/pflash0.img bs=1M count=64
dd if=/dev/zero of=./bootloader/pflash1.img bs=1M count=64
```

Luego podemos ejecutar u-boot especificando que bancos de memoria flash utilizar de la siguiente manera:

```
QEMU_AUDIO_DRV=none ./qemu-system-arm -m 128M -M vexpress-a9 \
    -nographic -kernel ./bootloader/u-boot/u-boot \
    -drive file=./bootloader/pflash0.img,if=pflash,format=raw \
    -drive file=./bootloader/pflash1.img,if=pflash,format=raw
```

Ahora es posible probar los comandos para guardar variables (`saveenv`) y los relacionados con la memoria flash (`flinfo`, `erase` y `protect`).

Para averiguar cual es la dirección de memoria dentro de la flash donde se almacenan las variables de entorno debemos consultar el archivo `vexpress_common.h`.

Ahí debemos buscar las definiciones de `CONFIG_ENV_ADDR` y de `CONFIG_ENV_OFFSET`.

Las mismas estan definidas como:

```
/* Store environment at top of flash */
#define CONFIG_ENV_OFFSET               (PHYS_FLASH_SIZE - \
                                        (2 * CONFIG_ENV_SECT_SIZE))
#define CONFIG_ENV_ADDR                 (CONFIG_SYS_FLASH_BASE1 + \
                                         CONFIG_ENV_OFFSET)
```

Vemos que la dirección base depende de `CONFIG_SYS_FLASH_BASE1` y de `CONFIG_ENV_OFFSET`. Además `CONFIG_ENV_OFFSET` depende de `PHYS_FLASH_SIZE` y `CONFIG_ENV_SECT_SIZE`.

Si buscamos dichas definiciones encontramos:

```
/* FLASH and environment organization */
#define PHYS_FLASH_SIZE                 0x04000000      /* 64MB */

#define CONFIG_SYS_FLASH_BASE1          V2M_NOR1

#define CONFIG_ENV_SECT_SIZE            FLASH_MAX_SECTOR_SIZE
```

y resolviendo las nuevas definiciones encontramos:

```
#define FLASH_MAX_SECTOR_SIZE           0x00040000      /* 256 KB sectors */

#define V2M_NOR1                (V2M_PA_CS1)
```

y a su vez:

```
#define V2M_PA_CS1              0x44000000
```

por lo tanto, la dirección de memoria flash donde se almacenaran las variables de entorno sera:

```
addr = 0x44000000 + 0x04000000 - 2 * 0x00040000 = 0x47f80000
```

Podemos ahora consultar dicha dirección de memoria a través de `u-boot` para ver su contenido:


```
md 0x47f80000

47f80000: 7de39244 68637261 6d72613d 75616200    D..}arch=arm.bau
47f80010: 74617264 38333d65 00303034 72616f62    drate=38400.boar
47f80020: 65763d64 65727078 62007373 6472616f    d=vexpress.board
47f80030: 6d616e5f 65763d65 65727078 62007373    _name=vexpress.b
47f80040: 5f746f6f 63735f61 74706972 616f6c3d    oot_a_script=loa
47f80050: 7b242064 74766564 7d657079 647b2420    d ${devtype} ${d
47f80060: 756e7665 243a7d6d 7369647b 5f6f7274    evnum}:${distro_
47f80070: 746f6f62 74726170 7b24207d 69726373    bootpart} ${scri
47f80080: 64617470 207d7264 72707b24 78696665    ptaddr} ${prefix
47f80090: 737b247d 70697263 203b7d74 72756f73    }${script}; sour
47f800a0: 24206563 7263737b 61747069 7d726464    ce ${scriptaddr}
47f800b0: 6f6f6200 66655f74 69625f69 7972616e    .boot_efi_binary
47f800c0: 2066693d 20746466 72646461 667b2420    =if fdt addr ${f
47f800d0: 615f7464 5f726464 203b7d72 6e656874    dt_addr_r}; then
47f800e0: 6f6f6220 69666574 6f6f6220 72676d74     bootefi bootmgr
47f800f0: 667b2420 615f7464 5f726464 653b7d72     ${fdt_addr_r};e
```

donde vemos que está el contenido de las variables de entorno que almacena `u-boot`.

Soporte para SD

También es posible simular un dispositivo tipo SD. Para ello, en primer lugar vamos a crear un archivo de 16M que represente el contenido de dicha memoria.

```
dd if=/dev/zero of=./bootloader/sd.img bs=1M count=16
```

y formateamos dicho archivo en `FAT 16`:

```
mkfs.vfat ./bootloader/sd.img
```

Luego debemos agregar a las opciones de inicio de `qemu` lo siguiente opción:

```
-drive file=./bootloader/sd.img,if=sd,format=raw
```

Es posible ahora probar los comandos relacionados con MMC/SD (como ser `mmc` y `mmcinfo`).

Portando u-boot a una nueva placa

Cada placa tiene archivos de configuracion en distintos directorios. Vamos a crear una nueva placa llamada `nova`, para ello vamos a seguir los siguientes pasos.

En primer lugar copiamos el directorio de la placa de la siguiente manera:

```
cp -R board/armltd/vexpress board/armltd/nova
```

Luego compiamos el archivo de configuracion existente de la placa de referencia `<nombre_placa_de_referencia>.h` como `<nombre_placa_personalizada>.h`, ejecutando:

```
cp include/configs/vexpress_ca9x4.h include/configs/nova.h
```

Dado que parte de la configuracion de la `vexpress` se encuentra en un archivo de cabecera `common`, tambien debemos copiarlo:

```
cp include/configs/vexpress_common.h include/configs/nova_common.h
```

Ademas debemos cambiar las definiciones de `__VEXPRESS_CA9X4_H` dentro del archivo `nova.h` y `__VEXPRESS_COMMON_H` dentro del archivo `nova_common.h`.

Creamos un archivo dentro de directorio `configs` con la nueva configuracion. Este puede ser una copia del de la placa de referencia:

```
cp configs/vexpress_ca9x4_defconfig configs/nova_defconfig
```

luego vamos a proceder a configurar dicho archivo.

Ahora renombramos:

```
mv board/armltd/vexpress/vexpress_common.c board/armltd/nova/nova_common.c
```

Modificamos el archivo `Makefile` del directorio `board/armltd/nova`, quedando solamente la linea:

```
obj-y  := nova_common.o
```

Cambiamos el archivo `Kconfig` del directorio `board/armltd/nova/Kconfig`:

```
if TARGET_NOVA

config SYS_BOARD
    default "nova"

config SYS_VENDOR
    default "armltd"

config SYS_CONFIG_NAME
    default "nova"

endif
```

Agregamos al archivo `Kconfig` del directorio `arch/arm/Kconfig` lo siguiente:

```
config TARGET_NOVA
    bool "Support nova"
    select CPU_V7A
    select PL011_SERIAL

source "board/armltd/nova/Kconfig"
```

Ahora podemos cargar nuestra configuracion:

```
make nova_defconfig
```

y ver que genere u-boot para nuestra nueva placa.

