Si, en vez de instalar OpenWrt en el router Banana Pi BPI-R3, queremos bootearle un Debian, en un ordenador con Debian ya instalado y ejecutándose, seguimos estos pasos:

Instalamos los siguientes paquetes:

```
sudo apt-get -y install bc binfmt-support binutils-aarch64-linux-gnu bison build-essential crossbuild-essential-arm64 debootstrap flex device-tree-compiler git gcc-aarch64-linux-gnu gcc-arm-linux-gnueabihf libncurses5-dev libncursesw5-dev libssl-dev make python3 python3-dev python3-pyelftools python3-setuptools qemu-user-static u-boot-tools
```

Clonamos el repositorio del kernel:

```
rm -rf /home/usuariox/debian-bpi-r3/linux
mkdir -p /home/usuariox/debian-bpi-r3 2> /dev/null
cd ~/debian-bpi-r3
git clone --depth 1 --branch "master" https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git
cd linux
```

Creamos una configuración personalizada del kernel:

```
make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- menuconfig
```

- Vamos a «Plattform selection» y marcamos solamente «Mediatek SoC family».

Salvamos los cambios al archivo con nombre .config. Ahora, si ejecutamos:

```
ls ~/debian-bpi-r3/linux/arch/arm64/boot/dts/mediatek | grep "bpi-r3"
```

La salida será algo así:

```
mt7986a-bananapi-bpi-r3.dts
mt7986a-bananapi-bpi-r3-emmc.dtso
mt7986a-bananapi-bpi-r3-nand.dtso
mt7986a-bananapi-bpi-r3-nor.dtso
mt7986a-bananapi-bpi-r3-sd.dtso
```

Ahí vemos que en el último kernel existen los archivos DTS (Device Tree) correspondientes a la BananaPi BPI-R3.

- Los archivos **.dts** (Device Tree Source) son archivos de texto que describen la estructura de hardware del sistema. Contienen información sobre los componentes de hardware, como CPUs, memoria, buses, dispositivos periféricos, etc.
- Los archivos **.dtso** (Device Tree Overlay Source) son sobreposiciones de Device Tree. Permiten definir cambios adicionales que se aplican sobre un Device Tree base. Son útiles para configurar hardware adicional o aplicar modificaciones sin cambiar el archivo principal.

Nos faltarían los archivos dtb y dtbo:

- Los archivos **.dtb** (Device Tree Blob) son la versión compilada (binaria) de los archivos .dts.
- Los archivos **.dtbo** (Device Tree Blob Overlay) son la versión compilada de los archivos .dtso. Estos archivos binarios pueden ser cargados en tiempo de ejecución para aplicar las configuraciones adicionales.

…por ello pasaremos a la línea de ejecución de compilación del kernel el parámetro dtbs. Entonces, compilamos el kernel ejecutando:

```
make -j$(nproc) ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- Image dtbs modules
```

Utilizamos debootstrap para crear un sistema de archivos raíz de Debian:

```
cd ~/debian-bpi-r3
```

```
sudo debootstrap --arch arm64 bookworm rootfs http://deb.debian.org/debian
```

Esto creará una estructura básica de Debian en el directorio rootfs.

Configuramos el sistema de archivos raíz

```
sudo mount -o bind /dev              /home/usuariox/debian-bpi-r3/rootfs/dev
sudo mount -o bind /sys              /home/usuariox/debian-bpi-r3/rootfs/sys
sudo mount -o bind /proc             /home/usuariox/debian-bpi-r3/rootfs/proc
sudo cp /usr/bin/qemu-aarch64-static /home/usuariox/debian-bpi-r3/rootfs/usr/bin/
sudo chroot /home/usuariox/debian-bpi-r3/rootfs
```

Dentro del chroot, configuramos los elementos básicos del sistema:

```
echo "bpi-r3" > /etc/hostname
cat <<EOF > /etc/apt/sources.list
deb http://deb.debian.org/debian bookworm main contrib non-free non-free-firmware
deb-src http://deb.debian.org/debian bookworm main contrib non-free non-free-firmware
deb http://deb.debian.org/debian-security/ bookworm-security main contrib non-free non-free-firmware
deb-src http://deb.debian.org/debian-security/ bookworm-security main contrib non-free non-free-firmware
deb http://deb.debian.org/debian bookworm-updates main contrib non-free non-free-firmware
deb-src http://deb.debian.org/debian bookworm-updates main contrib non-free non-free-firmware
EOF
apt-get update
apt-get -y install locales
apt-get -y install dialog
apt-get -y install sudo
dpkg-reconfigure locales
apt-get -y install linux-headers-$(uname -r)
apt-get -y install u-boot-tools
apt-get -y install initramfs-tools
exit
```

Copiamos el Kernel y los archivos .dtb al sistema de archivos raíz

```
sudo cp /home/usuariox/debian-bpi-r3/linux/arch/arm64/boot/Image /home/usuariox/debian-bpi-r3/rootfs/boot/
sudo cp /home/usuariox/debian-bpi-r3/linux/arch/arm64/boot/dts/mediatek/*bpi-r3*.dtb /home/usuariox/debian-bpi-
r3/rootfs/boot/
```

Preparamos los módulos, ejecutando:

```
cd /home/usuariox/debian-bpi-r3/linux/
sudo make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- modules_install INSTALL_MOD_PATH=/home/usuariox/debian-bpi-
r3/rootfs
```

Configuramos el bootloader para que pueda arrancar desde la SD ejecutando:

```
rm -rf /home/usuariox/debian-bpi-r3/u-boot 2> /dev/null
mkdir -p /home/usuariox/debian-bpi-r3 2> /dev/null
cd /home/usuariox/debian-bpi-r3
git clone --depth 1 --branch "master" https://source.denx.de/u-boot/u-boot.git/
cd u-boot
make -j$(nproc) CROSS_COMPILE=aarch64-linux-gnu- mt7986a_bpir3_sd_defconfig
make -j$(nproc) CROSS_COMPILE=aarch64-linux-gnu-
```

Creamos una nueva tabla de particiones en la tarjeta MicroSD, ejecutando:

```
sudo parted -s /dev/mmcblk0 mklabel gpt
```

Instalamos el bootloader en la tarjeta MicroSD ejecutando:

```
sudo dd if=/home/usuariox/debian-bpi-r3/u-boot/u-boot.bin of=/dev/mmcblk0 bs=512 seek=2
```

o

```
sudo dd if=/home/usuariox/debian-bpi-r3/u-boot/u-boot.bin of=/dev/sdg bs=512 seek=2
```

Creamos la partición ext4 en el espacio restante de la tarjeta MicroSD, iniciando desde 1 MiB (así dejamos sitio para el bootloader), ejecutando:

sudo parted -s /dev/sdg mkpart primary ext4 1MiB 100%

Formateamos la partición, ejecutando:

sudo mkfs.ext4 /dev/sdb1

Copiar Debian y el bootloader a la partición:

```
sudo mkdir -p /mnt/debianbpir3/
sudo mount /dev/sdb1 /mnt/debianbpir3/
sudo cp -R /home/usuariox/debian-bpi-r3/rootfs/* /mnt/debianbpir3/
sudo cp path/to/zImage /mnt/debianbpir3/boot/
sudo cp path/to/dtb /mnt/debianbpir3/boot/
sudo cp path/to/boot.scr /mnt/debianbpir3/boot/
sudo umount /mnt/debianbpir3/
```

sudo parted -s /dev/sdb mkpart EFI ext4 1MiB 1024MiB
sudo parted -s /dev/sdb mkpart OpenWrt ext4 1025MiB 28000MiB
sudo parted -s /dev/sdb mkpart Intercambio ext4 28001MiB 100%

# Formatear la partición para EFI como fat32
sudo mkfs -t vfat -F 32 -n EFI $vPrimerDisco»1″
# Formatear la partición para OpenWrt como ext4
sudo mkfs -t ext4 -L OpenWrt $vPrimerDisco»2″
# Formatear la partición para Intercambio como swap
sudo mkswap -L Intercambio $vPrimerDisco»3″

;;

4)

echo «»
echo » Marcando la partición EFI como esp…»
echo «»
sudo parted -s $vPrimerDisco set 1 esp on

Dentro de fdisk, realiza los siguientes pasos:

Crear una nueva partición primaria:
Elimina todas las particiones existentes (opcional pero recomendado).
Crea una nueva partición primaria (nueva partición 1).
Acepta los valores por defecto, lo que colocará la partición inmediatamente después del área ocupada por U-Boot.
Establece el tipo de sistema de archivos en Linux (83).

Escribe los cambios y sal.

Formatear la partición ext4:

sudo mkfs.ext4 /dev/sdX1

Copiamos el sistema operativo Debian

Monta la partición ext4 y copia el sistema operativo Debian, incluyendo el kernel (zImage o Image), el archivo de dispositivo (.dtb), y otros archivos necesarios para el arranque:

```
sudo mount /dev/sdX1 /mnt
# Copia los archivos necesarios a /mnt
sudo cp -R path/to/debian/rootfs /mnt/
sudo cp path/to/zImage /mnt/boot/
sudo cp path/to/dtb /mnt/boot/
sudo cp path/to/boot.scr /mnt/boot/
sudo umount /mnt
```

Creamos las particiones y formatear la tarjeta, ejecutando:

sudo fdisk /dev/sdX

Dentro de fdisk:
n (crear nueva partición)
p (primaria)
1 (número de partición)
<ENTER> (usar sector por defecto)
+100M (tamaño de la partición)
n (crear nueva partición)
p (primaria)
2 (número de partición)
<ENTER> (usar sector por defecto)
<ENTER> (usar el tamaño restante)
t (cambiar el tipo de la partición)
1 (seleccionar la partición 1)
c (tipo de partición FAT32)
w (escribir la tabla de particiones y salir)

# Formatear las particiones
sudo mkfs.vfat /dev/sdX1
sudo mkfs.ext4 /dev/sdX2

7. Copiar los Archivos al Sistema de Archivos

Monta las particiones y copia los archivos:

bash

sudo mount /dev/sdX1 /mnt
sudo cp -r BPI-R3-bsp/output/100MB/BPI-BOOT/* /mnt/
sudo umount /mnt

sudo mount /dev/sdX2 /mnt
sudo cp -r rootfs/* /mnt/
sudo umount /mnt

8. Configurar el Arranque

Edita los archivos de configuración de U-Boot en /mnt/ para que apunten al kernel y sistema de archivos correctos. Por ejemplo, asegúrate de que el boot.cmd (o boot.scr) esté configurado para cargar el kernel y rootfs desde las particiones correctas.
9. Finalizar y Probar

Desmonta todas las particiones, retira la tarjeta SD e insértala en tu BPI-R3. Conéctate a través de la consola serial para verificar el proceso de arranque.
Resumen

Prepara el entorno de construcción.
Descarga y compila el kernel.
Crea el sistema de archivos raíz de Debian.
Configura el sistema de archivos raíz.

Copia el kernel y los módulos.
Configura el bootloader.
Crea y formatea las particiones de la tarjeta SD.
Copia los archivos al sistema de archivos.
Configura el arranque.
Finaliza y prueba el arranque en el BPI-R3.

Siguiendo estos pasos, deberías poder crear una imagen de Debian oficial que pueda arrancar correctamente en tu router BPI-R3.