S13-ARM

Lenin G. Falconí

<2024-08-05 Mon>

Outline

Procesadores ARM

- Advanced RISC Machine
- Procesadores basados en una arquitectura RISC[1]
- Diseño simplificado, bajo consumo de energía
- Smartphones, tablets, portátiles, servidores

[1] Reduced Instruction Set Computer

Registros ARM

- Las operaciones del CPU se realizan en los registros del CPU (no en la memoria)
- Formada por 16 registros de 32 bits.
- R_0 a R_{13} son de propósito general
- R_{13}: stack pointer (llamada a funciones)
- R_{14}: Link register (llamada a funciones)
- R_{25}: Contador de Programa
- CSPR: registro que contiene información sobre la última instrucción ejecutada (e.g. overflow, negativo, 0, carry, etc.)

Formato Instrucción ARM

31-28	27-25	24-21	20
Condition	Operand Type	OpCode	Set Condition Codes

19-16	15-12	11-0
Operand Register	Destination Register	Inmediate Operand

Lenguaje Ensamblador

- El código carga el valor de 30 en el registro R_0 y 1 en el R_7. Luego usa SWI para terminar el programa generando una interrupción que pone al controlador en modo supervisor.

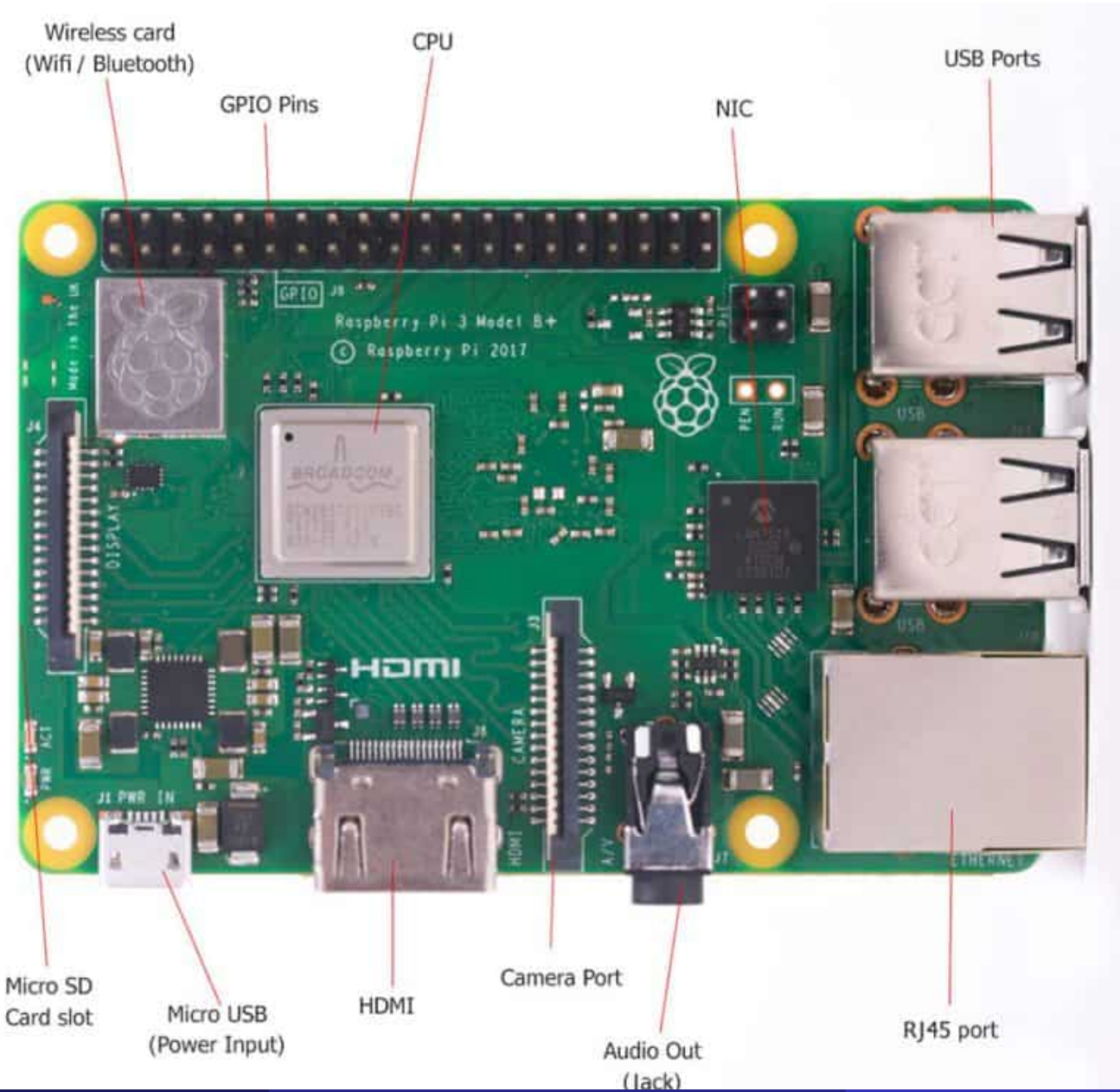
```
.global _start
_start:
mov R0, #30 /*destino, origen */
mov R7, #1  /*terminar programa*/
swi 0
```

- Para ingresar valores Hexadecimales se usa e.g. #0x0A

Raspberry PI

- La Raspberry Pi utiliza un system-on-chip (SoC) Broadcom BCM2835, que incluye una CPU compatible con ARM y una unidad de procesamiento gráfico (GPU) integrada.
- Los modelos originales de Raspberry Pi tenían un procesador de un solo núcleo que funcionaba a velocidades entre 700 MHz y 1.2 GHz.
- El sistema operativo predeterminado para Raspberry Pi se llama Raspberry Pi OS (anteriormente conocido como Raspbian)
- Raspberry Pi OS es una distribución basada en Linux diseñada específicamente para ejecutarse en todos los modelos de Raspberry Pi de 32 o 64 bits
- Alternativamente se pueden instalar otros Sistemas Operativos Linux o diseñar un kernel propio

Raspberry PI



Interfaces de Raspberry PI

- Interfaz con pines para entradas/salidas digitales
- Interfaz USB para conexión de dispositivos como teclados/mouse
- Salida HDMI para conexión de monitores/pantallas
- Jack de salida de sonido
- Las versiones actuales disponen de bus PCIe
- También dispone conectores para cámara web

Programación Assembler de Raspberry Pi

1. Creación de un archivo con extensión *.s
2. Compilación del archivo usando el GNU Assembler
3. Creación de archivo ejecutable

Programa Adder en Assembler de Raspberry Pi

Considere un programa denominado *adder.s* que sume dos valores cargados directamente a los registros w_1 y w_2 y cuyo resultado se almacena en el registro w_0. Se usa w en ve de r para trabajar en 32 bits.

```
.global
.func main
main:
mov w1, #0x0A
mov w2, #0x05
add w0, w1, w2
ret
```

Compilación y Ejecución

Se ensambla el programa adder.s en adder.o:

```
as -o adder.o adder.s
```

Se genera un ejecutable:

```
gcc -o adder adder.o
```

Se ejecuta y se enruta la salida a pantalla

```
./adder ; echo $?
```

Referencias:

- rpi.science
- bcm2711-arm-peripherals
- arm-isa
- video-intro-assembly-rp2040
- tutorial-assembly-ARM
- simulador