Semana 6

Explotación de BoF en ARM

ARM (Advanced RISC Machines)

- Esta presentación está basada en ARMv5
- Familia de CPUs que comparten principios de diseño similares y un instruction set común
- Bajo consumo de energía
- RISC (reduced instruction set computer)

> 💡 La idea es que la familia de CPU decide cuáles instrucciones van a soportar. Hay CPUs concretos que pertenecen a una misma familia, pero implementan distinto las instrucciones.

Principios de Diseño de CPUs RISC

- Instrucciones
- Pipelines
  - Un CPU no ejecuta las instrucciones en varios pasos.
  - Se hace un tipo de línea de producción.
- Registros
- Instrucciones carga-procesamiento-almacenamiento

> 💡 CPU más simple → frecuencia de reloj más alta.

Jerarquías de Memoria

- Eficiencia
- Velocidad
- vs Cantidad
- Se hace una línea logarítmica
  - Ahí se encuentran los registros, caché (Dentro del CPU)
  - RAM
  - Memoria Secundaria (Discos duros, almacenamiento en general)

💡 La arquitectura RISC le da énfasis al tema de los registros, incluso teniendo privilegios sobre ciertos registros.

ARM (Advanced RISC Machines)

▼ No es RISC “puro”

- 3 instruction set
  - 32 bits (original), 16 bits (thumb IS), 8 bits(Jazelle, el cual implementa la máquina virtual de Java en el CPU, lo cual mejora el rendimiento)
- Más instrucciones, e.g., digital signal processor
- Ciclos de ejecución para ciertas instrucciones > 1

> ARM trata de tomar las ventajas de un CISC y tener la velocidad y las ventajas de registros de los procesadores RISC

💡 ARM es muy popular debido a su bajo consumo eléctrico, lo cual lo llevó a estar presente en una variedad de dispositivos.

ARM: Sistemas Embebidos

- Hardware Device: Es el que soporta todo lo que se explica en esta lista.
- Inicialización: Se cargan las primeras instrucciones para arrancar el dispositivo. Al tener energía, un dispositivo siempre tendrá un reset state. Generalmente cuenta con una ROM para ello.
- Device drivers: Son los controladores del sistema que se encargan de controlar las entradas y salidas del sistema.
- Sistema Operativo: generalmente es un calendarizador de aplicaciones.
- Aplicaciones: Software final que se usa en el sistema.

> 💡 Una de las funciones principales del SO es tener un schedule, el cual prioriza a quién se le da la siguiente instrucción.

ARM registros

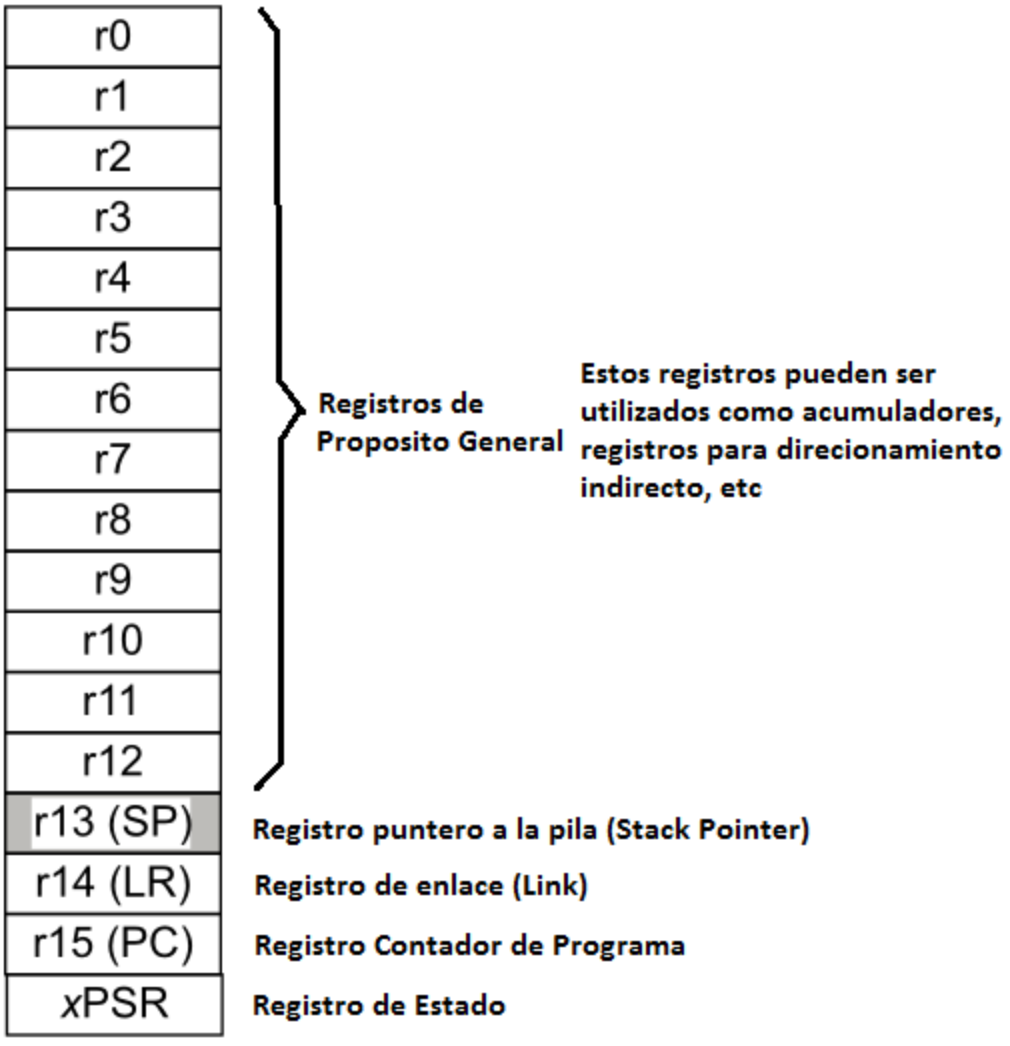
- Almacenan datos o direcciones
- Registros disponibles en user mode: r0- r12,
  - r13: stack pointer
  - r14: link register (return address)
  - r15: program counter(puntero a la siguiente instrucción por ejecutar)
  - cpsr, spsr (program status register) → (negativos, ceros, overflow aritmético, flags en general)

ARM: Modos de Operación

- Privilegiados
  - Abort
  - Fast Interrupt request
  - Interrupt request

    - supervisor
    - system
    - undefined
- No privilegiados
    - User

Todos los registros de ARM



ARM: Linux

- Registros disponibles en user mode

- r0-r10 (r0-r3 se encargan de los argumentos de las funciones)
- r11: frame pointer
- r12: intra-procedure: para hacer llamadas de una rutina a una subrutina

ARM: Ensamblador

- Clases de Instrucciones
  1. Branch
  2. Data-processing
  3. Status register transfer
  4. Load and Store
  5. Coprocessor
  6. Excepion-generating
- Ejecución Condicional

Mnemonic	Name	Condition flags
EQ	equal	Z
NE	not equal	z
CS HS	carry set/unsigned higher or same	C
CC LO	carry clear/unsigned lower	c
MI	minus/negative	N
PL	plus/positive or zero	n

Branch

- B: Branch
- BL: Branch with link
- BX: Branch with Exchange
- BLX: Branch with...
- Manipulación directa del r15!

B: subroutine: unconditional jump

BEQ: subroutine: conditional jump

BL: subroutine: function call

Data Processing

- Logical
  - AND
  - ORR
  - EOR
- Arithmetic
  - SUB
  - ADD
- Comparison
  - COMP
- Test
  - Sirve para comparar dos veces, es decir, hace un AND

Status Register Transfer

- MRS: Move Status Register yo General Purpose Registerr
- MSR: viceversa
- Útil para:
  - Cambiar estado
  - Cambiar modo
  - Cambiar endianness

Load and Store

- LDR: Load Word
- LDRB: Load Byte
- SRT: Store Word

- STRB: Store Byte

Coprocessor

- CDP: Coprocessor data operation

Exception-generating

- BKPT: breakpoint
- SWI: Software interrupt
- A veces el software que utilizamos necesita realizar ciertas llamadas al sistema.

BoF Defenses

> 💡 BoF es independiente de la arquitectura y el dispositivo.

BoF Attacks

- Stack
  - Jump to deadcode
  - Jump to libc
  - Shellcode
- Heap
  - Data corruption

Stack Canary

> 💡 Antes a las minas se llevaban un canario dado que son extremadamente sensibles para ciertos gases, lo cual también es perjudicial para los humanos. Si el pájaro moría, era momento de irse

- Value stored before (lower in memory) the saved return address
- At runtime, it is checked if the canary value has changed

> 💡 Se trata de corroborar que un valor canario no se haya cambiado antes de saltar a la dirección de retorno.

> 💡 El valor del canario siempre es conocido y siempre tiene un valor 00 dentro de los 32 bits, dado que el 00 denota el cierre del string.

Los compiladores modernos típicamente tienen este feature activado por defecto.

Para desactivarlo se usa:

```
gcc ... -fno-stack-protector
```

¿Qué ataque puede tener el canario?

- Ataque probabilístico
- Tratar de averiguar el canario y no corromperlo