ÚRAD PODPREDSEDU VLÁDY SR
PRE INVESTÍCIE
A INFORMATIZÁCIU

CSIRT.SK

Architektúra ARM

Mgr. Ján Kotrady

About this course

- Introduction to Android reverse engineering
- Introduction to ARM architecture
- Beta course

+421 2 59 278 262
jan.kotrady@csirt.sk

Štefánikova 15 - 011 05 Bratislava
www.csirt.gov.sk

Windows malware analysis (diff)

- ssdeep
- procmon / procexp
- PE files
- .dll files, exports, imports
- Entropy
- Strings
- Yara rules

+421 2 59 278 262
jan.kotrady@csirt.sk

Štefánikova 15 - 011 05 Bratislava
www.csirt.gov.sk

APK generally

- Android applications
- Dalvik
- Java
- .so files


```
public MainActivity() {
  super();
  currentPosition = 0;
}
```

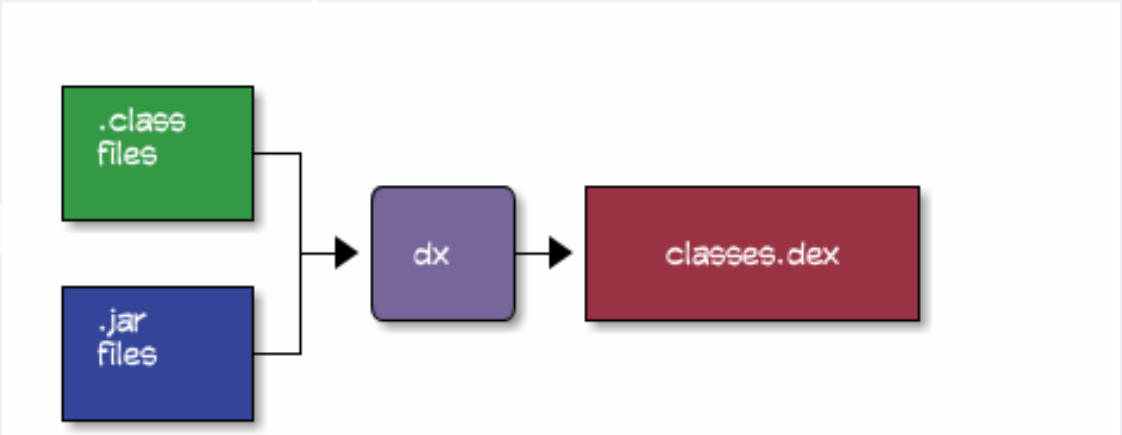
Source: https://github.com/dogriffiths/HeadFirstAndroid/wiki/How-Android-Apps-are-Built-and-Run

+421 2 59 278 262
jan.kotrady@csirt.sk

Štefánikova 15 - 011 05 Bratislava
www.csirt.gov.sk

Conversion to Dalvik code

- .class file contains standard Java byte-codes
- Android uses a Dalvik (until 4.4 – 5.0), replaced with ART
- Dalvik is a Virtual environment



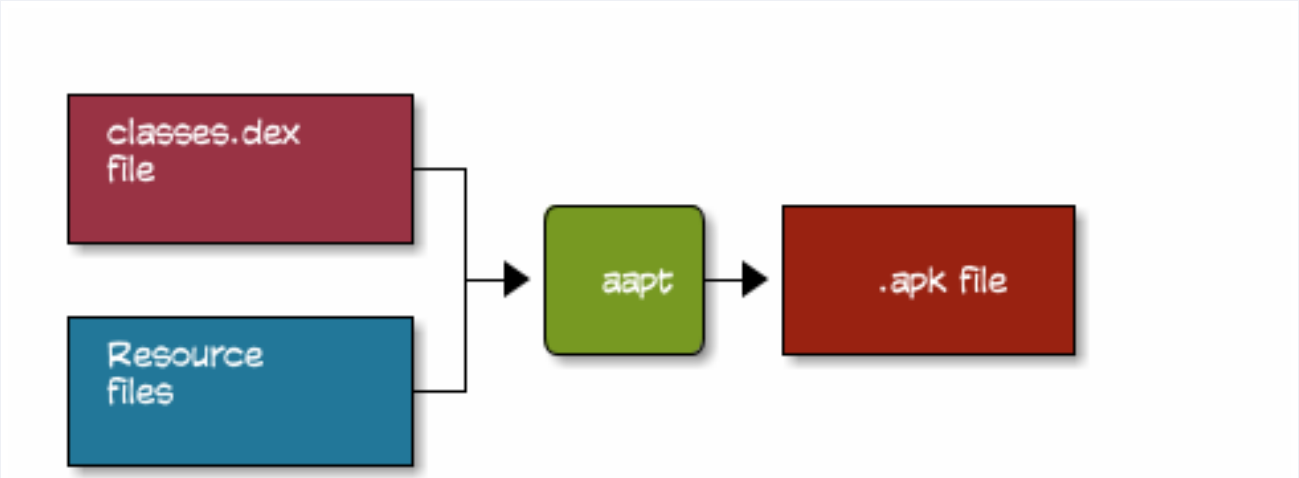
Source:https://github.com/dogriffiths/HeadFirstAndroid/wiki/How-Android-Apps-are-Built-and-Run

+421 2 59 278 262
jan.kotrady@csirt.sk

Štefánikova 15 - 011 05 Bratislava
www.csirt.gov.sk

Dalvik dex -> apk file

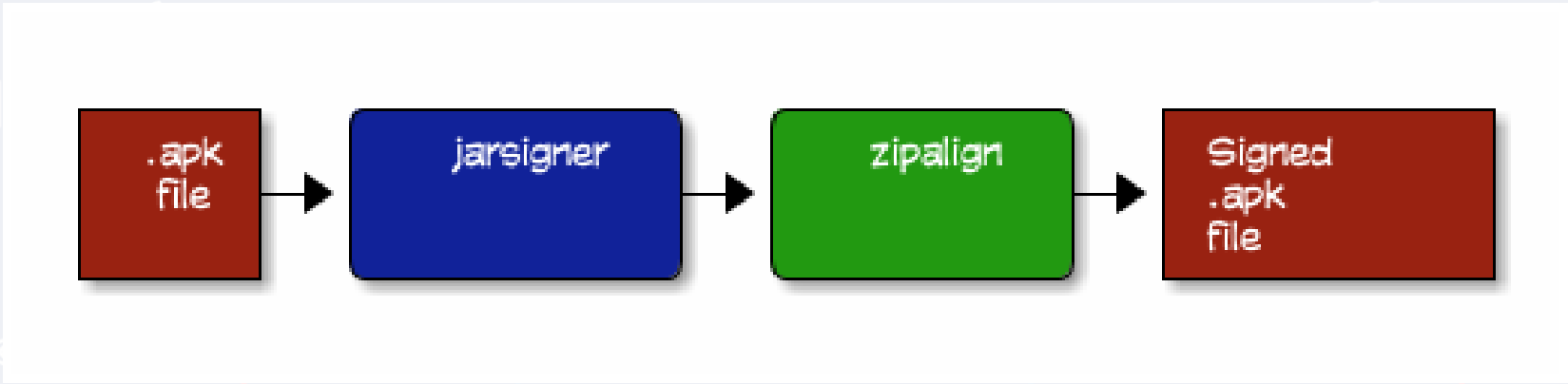
- APK file (.apk) is a “zip” file
- *Android Asset Packaging Tool (aapt)*



Source:https://github.com/dogriffiths/HeadFirstAndroid/wiki/How-Android-Apps-are-Built-and-Run

Signing android app

- Mobile devices don’t support applications without signatures
- jarsigner



Source:https://github.com/dogriffiths/HeadFirstAndroid/wiki/How-Android-Apps-are-Built-and-Run

Running applications

- adb transfers app to a mobile device
- Android fork Zygote
  A fork is a classic Linux command.
- After Zygote start, it converts .dex file into elf Linux file
  Dex2oat
  /data/dalvik-cache/[cpu version]/….
- And app will load native library

+421 2 59 278 262
jan.kotrady@csirt.sk

Štefánikova 15 - 011 05 Bratislava
www.csirt.gov.sk

Native library

- Do you know, what is a Dynamic load library?
- What are the equivalents for Linux?

+421 2 59 278 262
jan.kotrady@csirt.sk

Štefánikova 15 - 011 05 Bratislava
www.csirt.gov.sk

Native library

- Do you know, what is an Dynamic load library ?
- What are Linux equivalents ?
- .so files
- They are “same” as a Linux shared library, but with little changes …
- C/C++ source code
- .a – static library
- .so – shared library (shared objects)
- Glibc is a Linux library, Android uses Bionic C library

+421 2 59 278 262
jan.kotrady@csirt.sk

Štefánikova 15 - 011 05 Bratislava
www.csirt.gov.sk

Native library

- ELF files
- Different headers as in a standard PE files
- Readelf, elfdump, objdump
- ELF HEADER

  EI_CLASS – 32 or 64 bits architecture

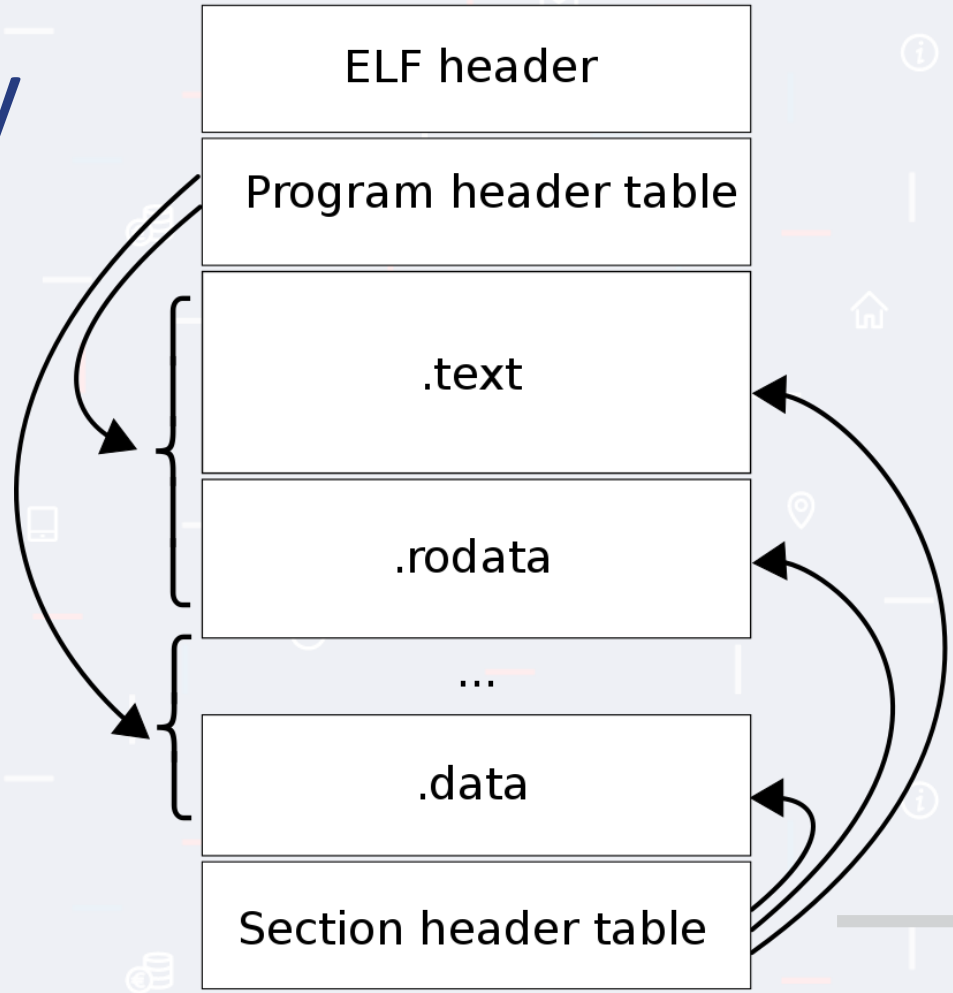
  EI_DATA – endianness

  E_machie – x86, PowerPC, ARM, AArch64

  E_entry – entry point
- Program header

  Filesize, virual address, offset, memsize

  P_load – how big chunk will be fetch into memory



source: wiki ☺

Native library

- Section header specifies sections
- Program header specifies segments
  Rights in memory
  Memory locations
- Sections specify which data are in the memory.
  Formatted data
  Instructions
  Data important for linkers

+421 2 59 278 262
jan.kotrady@csirt.sk
Štefánikova 15 - 011 05 Bratislava
www.csirt.gov.sk

Native library

- Sections:

  init – initialization data
  .finit – flip of init ☺
  .text – main program part
  .bss – structures
  .data – data part, writable
  .rodata – read-only data
  .got + .plt – import table in x86
  readelf -D -s ./file.so – reading export tables from .dynsym

+421 2 59 278 262
jan.kotrady@csirt.sk
Štefánikova 15 - 011 05 Bratislava
www.csirt.gov.sk

Android native library

- Different systems calls, Bionic C
- Little bit different linking as in Linux
- Android JNI
- https://docs.oracle.com/javase/7/docs/technotes/guides/jni/spec/functions.html

```
JNIEXPORT jstring JNICALL
Java_com_example_hellojni_HelloJni_stringFromJNI( JNIEnv* env,
                                                  jobject thiz )
```

```
JNIEXPORT jdouble JNICALL Java_your_package_structure_className_myMethod
(JNIEnv * env, jobject jobj, jstring path) {
     char * path;

    path = (*env)->GetStringUTFChars( env, path , NULL ) ;
```

GetStringUTFLength

```
jsize GetStringUTFLength(JNIEnv *env, jstring string);
```

Returns the length in bytes of the modified UTF-8 representation of a string.

LINKAGE:
Index 168 in the JNIEnv interface function table.

+421 2 59 278 262
jan.kotrady@csirt.sk

Štefánikova 15 - 011 05 Bratislava
www.csirt.gov.sk

Adb

- Android debug bridge
- Used for installing applications

  adb devices – list all available devices

  adb tcpip 5555 – run adp listener

  adb install – install application

  adb uinstall – guess ☺

  adb push – upload files

  adb pull – guess ☺

  SMS data: /data/data/com.android.providers.telephony/databases

+421 2 59 278 262
jan.kotrady@csirt.sk

Štefánikova 15 - 011 05 Bratislava
www.csirt.gov.sk

Android files and resources

- AndroidManifest.xml

  Every android applications have one at project root

  Contains:

  Permissions, App package name, Android versions, Activity – like windows in desktop app

- Res

  Layouts, buttons text, text-field texts etc.

  Settings for text-fields

- Lib

  .so files

+421 2 59 278 262
jan.kotrady@csirt.sk

Štefánikova 15 - 011 05 Bratislava
www.csirt.gov.sk

Android permissions

- Normal permissions

  Listed in the manifest file

  Autofill, nfc, network, text, voice interaction, install packages, write settings …
- Dangerous permissions

  User must agree from android 6.0

  Below 6.0 – Dangerous permissions are asked while app is installed

  read_calendar, camera, record_audio, read_call_log, send_sms
- Android group permissions

  READ_SMS, SEND_SMS -> allow one, allow both …

+421 2 59 278 262
jan.kotrady@csirt.sk

Štefánikova 15 - 011 05 Bratislava
www.csirt.gov.sk

Services and broadcast listeners

- Services – long running tasks

  Foreground, background, Bound service (client/server)

  Declared in the manifest file

- Broadcast – used for communications

  Can be used between services and Activity

  Also, communications between applications.

  In the manifest file as well

+421 2 59 278 262
jan.kotrady@csirt.sk

Štefánikova 15 - 011 05 Bratislava
www.csirt.gov.sk

```
<manifest ... >
  ...
  <application ... >
      <service android:name=".ExampleService" />
      ...
  </application>
</manifest>
```

```
<receiver android:name=".MyBroadcastReceiver"  android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED"/>
        <action android:name="android.intent.action.INPUT_METHOD_CHANGED" />
    </intent-filter>
</receiver>
```

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="com.example.snazzyapp">

    <uses-permission android:name="android.permission.SEND_SMS"/>

    <application ...>
        ...
    </application>
</manifest>
```


+421 2 59 278 262
jan.kotrady@csirt.sk

Štefánikova 15 - 011 05 Bratislava
www.csirt.gov.sk

XPOSED + INSPECKAGE

- Applications injections.
- Rewriting instructions in applications.
- Required root permissions.
- Easy to use.
- Nice gui.

+421 2 59 278 262
jan.kotrady@csirt.sk

Štefánikova 15 - 011 05 Bratislava
www.csirt.gov.sk

Frida

- Same as the INSPECKAGE
- Hard to use, easy to install
- Javascript programming
- adb push frida-server /data/local/tmp/
- adb shell “chmod 755 /data/local/tmp/frida-server”
- adb shell “/data/local/tmp/frida-server &”
- adb devices –l
- frida-ps -U

+421 2 59 278 262
jan.kotrady@csirt.sk

Štefánikova 15 - 011 05 Bratislava
www.csirt.gov.sk

Frida

- python for loading
- Javascript for interception

```
Import Frida
session = frida.get_usb_device().attach(“app package”)
script = session.create_script([script_file])
Script.load()
```


+421 2 59 278 262
jan.kotrady@csirt.sk

Štefánikova 15 - 011 05 Bratislava
www.csirt.gov.sk

Frida

- Script (js) for Frida:

```
var Activity = Java.use(“activity”)
Activity.[functions].implementation = function(x,y) {
return [some function](…);
}
```


+421 2 59 278 262
jan.kotrady@csirt.sk

Štefánikova 15 - 011 05 Bratislava
www.csirt.gov.sk

Emulators and tools

- Apktool
- Santoku
  https://www.nowsecure.com/tools-and-trainings/forensics-community-edition/
  https://santoku-linux.com/download/
  Viaforensics, Nowsecure
  Username/password: santoku / santoku123
- Genymotion
- AVD

+421 2 59 278 262
jan.kotrady@csirt.sk

Štefánikova 15 - 011 05 Bratislava
www.csirt.gov.sk

Emulators and tools

- Radare2 – cutter
- Rock64

  Install and run android 8/9

  ARM architecture

  AArch64 and ARMv8

  Cortex A53

+421 2 59 278 262
jan.kotrady@csirt.sk

Štefánikova 15 - 011 05 Bratislava
www.csirt.gov.sk

ARM – Basic Features

- RISC architecture
  Small compared to x86
- More general purpose registers
- Fixed instruction length: (16 bits or 32 bits)
- Load – store mode
  Data must be moved from memory into registers before operated on
  Only load-store instruction can operate memory
  LDR/STR instruction

+421 2 59 278 262
jan.kotrady@csirt.sk
Štefánikova 15 - 011 05 Bratislava
www.csirt.gov.sk

ARM – ARM/Thumb

- Mad ☹
- Define instruction set
- ARM

  32 bits instruction length - always
- Thumb

  16 or 32 bits instruction length

  .w - stands for 32 bits
- Determining mode:

  If jump address' LSB is 1 => Thumb

  If T bit in CPSR register (EEFLAGS) is set to 1 => Thumb

+421 2 59 278 262
jan.kotrady@csirt.sk
Štefánikova 15 - 011 05 Bratislava
www.csirt.gov.sk

ARM – ARM/Thumb/Thumb2/A64/A32

- ARM/THUMB from ARMv4T to ARMv7-A
- ARMv6T2 – Thumb 2 – adds more 32b instructions to Thumb

  UAL – replaces separate ARM and Thumb syntax to allow writing code once and assembling it to either instruction set without modification

  Adds IT conditional jump
- Try to reduce the confusion

  ARMv8

  AArch32 – Thumb, Thumb 2, ARM

  AArch64 – Same instruction size as AArch32, operate on both 32b reg. or 64b reg.

+421 2 59 278 262
jan.kotrady@csirt.sk

Štefánikova 15 - 011 05 Bratislava
www.csirt.gov.sk

ARM – Data types and registers

- Byte: 8-bit (-b)
- Half-word: 16-bit (-h)
- Word: 32-bit
- Double-word: 64-bit
- 16x word registers (general purpose)

  R0,…,R15

  R13 – SP (ESP)

  R14 – Link register LR – return address – will be discussed later

  R15 – PC (EIP) – pointing to ??next instruction??

  R7 – holds system call number (SWI, SVC instructions)

ARM – R15

- Address of current instruction + 8 (2 ARM instruction) or +4 in thumb

  Q: What is the difference from x86 ?

  Q: + 8 is a two arm instruction. Why +8?

- We can write and read from R15 (SP)

  Writing means: directly execute instruction at R15

  But we must respect the rings!

+421 2 59 278 262
jan.kotrady@csirt.sk

Štefánikova 15 - 011 05 Bratislava
www.csirt.gov.sk

ARM – Intro to instruction set

- LDR/STR
  - Load register
  - Store register
  - LDR vs MOV
    - Sometimes, it is possible to use mov, but it‘s inefficient
    - mov has smaller address space
    - Depends on the architecture (there isn’t only 1 ARM type ☺)
    - Always use LDR
  - 1,2,4 bytes to and from memory

+421 2 59 278 262
jan.kotrady@csirt.sk

Štefánikova 15 - 011 05 Bratislava
www.csirt.gov.sk

ARM – Pre-index & post-index

- LDR/STR

  ! - mark

  Base register is updated with the final memory address used in the reference operation

```
12 F9 01 3D    LDRSB.W R3, [R2 ,#-1]! ; R3 = *(R2-1)
                                      ; R2 = R2-1

10 F9 01 6B    LDRSB.W R6, [R0],#1    ; R6 = *R0
                                      ; R0 = R0+1
```


+421 2 59 278 262
jan.kotrady@csirt.sk

Štefánikova 15 - 011 05 Bratislava
www.csirt.gov.sk

ARM – LDM/STM

- Multiple register store
- STM

  STM R1, {R6-R10}

  R6 at R1, R7 at R1+4, R8 at R1+8 etc.
- Update base register

  ! – update mark

  STM R1!, {R6-R10}

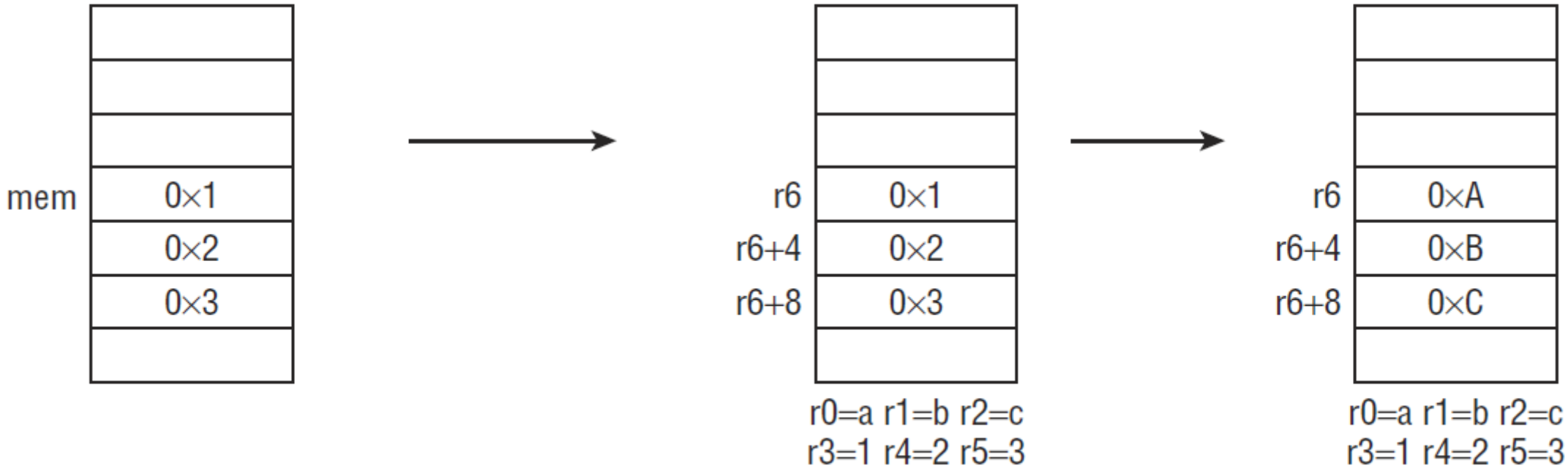
  R1 is updated immediately after R10 is stored

+421 2 59 278 262
jan.kotrady@csirt.sk

Štefánikova 15 - 011 05 Bratislava
www.csirt.gov.sk

```
ldr r6, =mem
mov r0, #10
mov r1, #11
mov r2, #12
ldm r6, {r3, r4, r5}
```

```
stm r6, {r0, r1, r2}
```



ARM – Function calls

- No calls as in x86 (and x86-64)
- Return address in stack or in LR – link register
- Jump can switch between ARM and Thumb state – depends on destination
- First four 32bit parameters are passed via R0-R3, rest on the stack

  ?? RIGHT TO LEFT ?? – No relevant source
- Return is stored in R0 (-R3)
- R4-R11 – calle save – What does it mean?

+421 2 59 278 262
jan.kotrady@csirt.sk

Štefánikova 15 - 011 05 Bratislava
www.csirt.gov.sk

ARM – Function calls - jumps

- B, BX, BL, BLX
- B

  Never return function

  Same as jmp in x86

  Transfer control

  B imm ; imm – relative offset
- BX – branch and exchange

  Can switch between ARM and THUMB (LSB 1, then THUMB)

  BX <register>

  Usually return from a function (same as RET)

+421 2 59 278 262
jan.kotrady@csirt.sk

Štefánikova 15 - 011 05 Bratislava
www.csirt.gov.sk

ARM – Function calls - jumps

- B, BX, BL, BLX
- BL – branch with link

  Store return address in LR

  call

  Invoke functions

  Cant change ARM or THUMB

  Takes only offset!
- BLX – branch with link and exchange

  Can switch between ARM and THUMB (LSB 1, then THUMB)

  BX <register> (or offset)

+421 2 59 278 262
jan.kotrady@csirt.sk

Štefánikova 15 - 011 05 Bratislava
www.csirt.gov.sk

ARM – Conditional run Thumb state

- No conditions
- Use with IT

  4 instructions on condition

  Syntax IT xyz cc

    cc – condition for first instruction

    x,y,z condition for second, third, fourth

      Can be: T or E

      T – match CC

      E – not match CC

- Switch case

  Many conditions ☺

+421 2 59 278 262
jan.kotrady@csirt.sk

Štefánikova 15 - 011 05 Bratislava
www.csirt.gov.sk

```
ITTE   NE           ; Next 3 instructions are conditional
ANDNE  R0, R0, R1   ; ANDNE does not update condition flags
ADDSNE R2, R2, #1   ; ADDSNE updates condition flags
MOVEQ  R2, R3       ; Conditional move

ITE    GT           ; Next 2 instructions are conditional
ADDGT  R1, R0, #55  ; Conditional addition in case the GT is true
ADDLE  R1, R0, #48  ; Conditional addition in case the GT is not true

ITTEE  EQ           ; Next 4 instructions are conditional
MOVEQ  R0, R1       ; Conditional MOV
ADDEQ  R2, R2, #10  ; Conditional ADD
ANDNE  R3, R3, #1   ; Conditional AND
BNE.W  dloop        ; Branch instruction can only be used in the last instruction of an IT block
```

Zdroj: https://azeria-labs.com/arm-conditional-execution-and-branching-part-6/

+421 2 59 278 262
jan.kotrady@csirt.sk

Štefánikova 15 - 011 05 Bratislava
www.csirt.gov.sk

ARM – Arithmetic

- Add, sub, mul, and, orr
- NO DIV! (only pseudo by the standards) (actually, in ARMv7-R and ARMv7-M there is a DIV instruction)
- ARMv8 ? Newest ?
- .s suffix – set flags!

```
01: 4B 44          ADD     R3, R9              ; r3 = r3+r9
02: 0D F2 08 0B    ADDW    R11, SP, #8         ; r11 = sp+8
03: 04 EB 80 00    ADD.W   R0, R4, R0,LSL#2    ; r0 = r4 + (r0<<2)
04: EA B0          SUB     SP, SP, #0x1A8      ; sp = sp-0x1a8
```


+421 2 59 278 262
jan.kotrady@csirt.sk

Štefánikova 15 - 011 05 Bratislava
www.csirt.gov.sk

Thanks for paying attention.