BitaBit - Programación a bajo nivel

•••

Franco Catrin 2020

Programación de **ALTO** nivel (de abstracción)

```
Código de Aplicación
  Lenguajes / SDK (Java, C, PHP)
APIs (win32, libc, OpenGL, DirectX)
Sistema Operativo (Android, Linux)
Código de máquina (6502, Z80, x86)
            Hardware
```

```
3⊕ import org.w3c.dom.Element;
11
   public class RectangleWidget extends Widget {
        private Background background;
13
14
150
        public RectangleWidget(Theme theme. Element node) {
16
            super(theme, node);
17
            background = Background.fromNode(theme, node);
18
19
200
21
        public View createView(Context context. ViewGroup parent) {
22
            View view = new View(context);
23
            parent.addView(view);
24
            applyBounds(view);
25
            background.apply(view);
26
            return view:
27
28
29 }
30
```

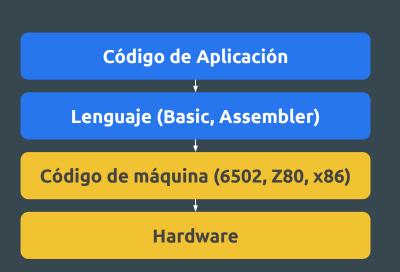
Ejemplo de Código de Máquina 6502

Código de máquina (6502, Z80, x86)

Hardware

```
BLOCK 1 patched_screaming_wings.xex
  B700: A9 80
                  lda #$80
                                  : DOSVEC
  B702: 85 0A
                  sta $0A
  B704: 85 0C
                  sta $0C
                                  ; DOSINI
  B706: A9 43
                  lda #$43
  B708: 85 0B
                  sta $0B
                                  : DOSVEC+1
  B70A: 85 0D
                  sta $0D
                                  : DOSINI+1
  B70C: 18
  B70D: A2 01
                  ldx #$01
  B70F: 86 09
                                  ; B00T
                  stx $09
  B711: CA
                  dex
  B712: 8E 44 02
                                  : COLDST
                  stx $0244
6 B715: BD 62 B7
                  lda $B762.x
                                  ; L_B75E+4
  B718: E9 6F
                  sbc #$6F
18 B71A: 9D 62 B7
                  sta $B762,x
                                  ; L B75E+4
  B71D: E8
  B71E: E0 64
                  cpx #$64
  B720: D0 F3
                                  ; L_B715
                  bne $B715
  B722: A9 41
                  lda #$41
  B724: 8D 02 D4
                  sta $D402
                                  ; DLISTL
  B727: 8D 30 02
                  sta $0230
                                  ; SDLSTL
                  lda #$B7
  B72A: A9 B7
26 B72C: 8D 03 D4
                  sta $D403
                                  : DLISTH
  B72F: 8D 31 02 sta $0231
                                  : SDLSTH
  B732: A2 03
                  ldx #$03
  B734: BD 5E B7 lda $B75E.x
                                  : L B75E
  B737: 9D 16 D0
                  sta $D016,x
                                  ; COLPFO
31 B73A: 9D C4 02
                  sta $02C4.x
                                  ; COLORO
32 B73D: CA
                  dex
  B73E: 10 F4
                  bpl $B734
                                  : L B734
84 B740: 60
                  rts
                  BYTE $70 : DL 8 scanlines
85 B741: 70
                  BYTE $70 : DL 8 scanlines
86 B742: 70
  B743: 70
                  BYTE $70 : DL 8 scanlines
```

Programación de **BAJO** nivel



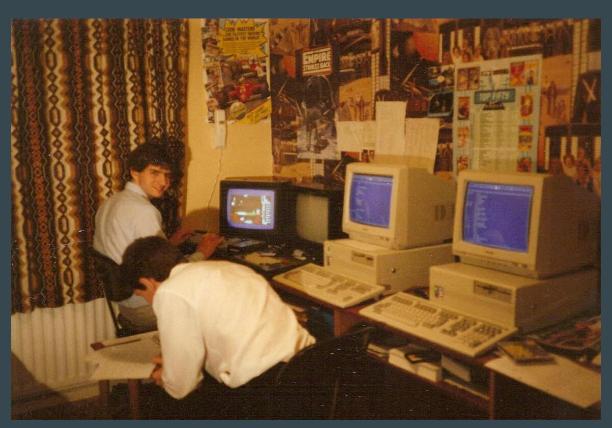
```
ENEMY SETS1
                 cmp #$00
                 bne ENEMY SETS2
                 lda ENEMY TBLS1,x
                 sta ENEMY_TYPE
                 rts
ENEMY_SETS2
                 cmp #$01
                 bne ENEMY SETS3
                 lda ENEMY_TBLS2+1,x
                 sta ENEMY TYPE
                 rts
ENEMY_SETS3
                 CMD #$02
                 bne ENEMY SETS4
                 lda ENEMY_TBLS3+2,x
                 sta ENEMY TYPE
                 rts
ENEMY SETS4
                 lda ENEMY TBLS4+3.x
                 sta ENEMY TYPE
                 rts
ENEMY_TBLS1
                 .byte $01, $02, $03, $01, $03, $02, $04, $05
                 .byte $05, $01, $01, $02, $04, $03, $04, $01
ENEMY_TBLS2
                 .byte $03, $01, $05, $05, $06, $04, $07, $05
ENEMY_TBLS3
                  .byte $02, $04, $01, $0A, $0B, $02, $01, $0B
                  byte $08, $09, $05, $05, $03, $0A, $02, $07
ENEMY_TBLS4
                 byte $08, $07, $03, $06, $07, $08, $09, $0A
                  .byte $0B. $05. $05. $08. $09. $02. $03. $0A
```

¿Por qué aprender a programar a bajo nivel?

- ¡Es divertido!
- Presenta desafíos de programación únicos
- Mejora el conocimiento de los computadores y su funcionamiento
 - Es más fácil encontrar y entender errores
 - Es más fácil entender optimizaciones
 - Es más fácil entender nuevas tecnologías
- Permite aprender a programar cualquier cosa, incluso sin herramientas.

¡Ah, y también se pueden crear nuevos juegos para sistemas antiguos!!

Make **Bedroom Coding** great again



¿Qué se necesita?

- Entusiasmo
- Café o equivalente
- Conocimientos de programación son deseables pero no necesarios

Recomendaciones e indicaciones (1 de 2)

- Es normal si no entiendes algo a la primera
- Con el tiempo las piezas van encajando
- Puedes dejar pasar algunas sesiones antes de meter las manos
- Habrán sesiones en vivo para preguntas
- Habrán links o videos de apoyo para:
 - Instalación y configuración de emuladores
 - o Instalación y configuración de ensambladores
 - o Instalación y configuración de editores o ambientes de desarrollo

Recomendaciones e indicaciones (2 de 2)

- No es un curso de Assembler
 - Hay cursos especializado que puedes ver en paralelo (DogDark, Guillermo Fuenzalida)
- En momentos se sacrificará exactitud para mejorar explicaciones
- Algunos temas los iremos aprendiendo juntos en el camino
- El enfoque será práctico
 - Se crearán un par de juegos
 - Se enseñará lo necesario para hacerlos
 - Para profundizar los conocimientos hay otros canales.

Plan de trabajo inicial

- Temporada 1: Programación a bajo nivel en **Atari 800**
 - Assembler **6502**, gráficos y sonido
- Temporada 2: Programación a bajo nivel en ZX Spectrum
 - Assembler Z80, gráficos y sonido
- Temporada 3: Programación a bajo nivel en MSX/MSX2
 - Assembler **Z80**, gráficos y sonido
- Temporada 4: Programación a bajo nivel en PC / MSDOS
 - Assembler **x86**, gráficos y sonido

Ahora vé por tu café!