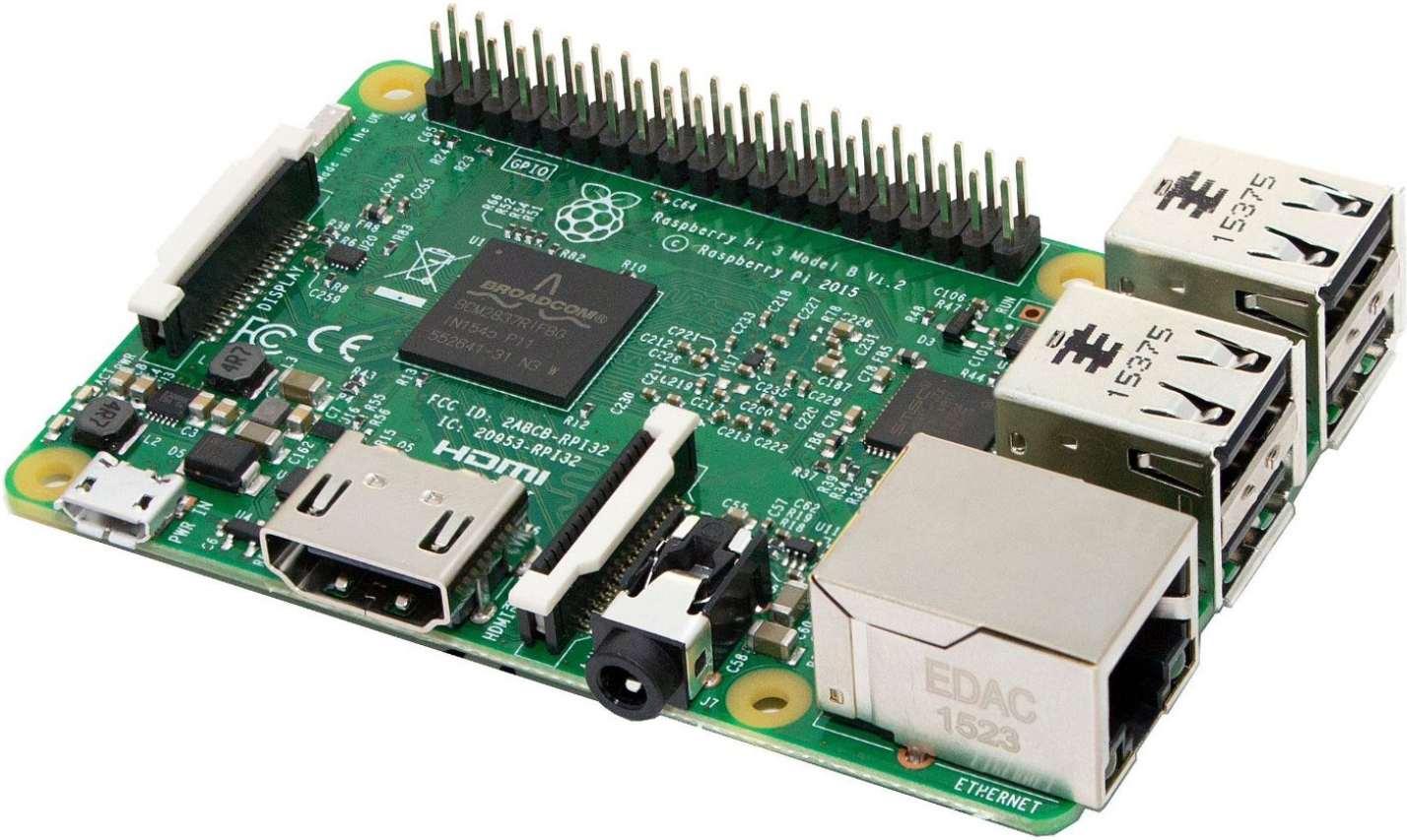
Etude d’un CPU inconnu

Présenté 23/06/2017

Pour BeeRumP 2017

Par Fabien Perigaud

Etude d’un CPU inconnu



License keys

- **Decoding hardware**
  - MPEG2
  - VC1

Dépendances

Product Description

This key will enable a single Raspberry Pi to decode MPEG-2 video in hardware. You will need to provide your device's internal 16-digit serial number as part of your order. **Your serial number is not the number printed on your board.**

To find your serial number, type **cat /proc/cpuinfo** at the command line as shown below:

```
pi@raspberrypi:~$ cat /proc/cpuinfo
Processor       : ARMv6-compatible processor rev 7 (v6l)
BogoMIPS        : 697.95
Features        : swp half thumb fastmult vfp edsp java tls
CPU implementer : 0x41
CPU architecture: 7
CPU variant     : 0x0
CPU part        : 0xb76
CPU revision    : 7

Hardware        : BCM2708
Revision        : 1000002
Serial          : 000000000000000d
```

Installation

- **Fichier /boot/config.txt**

  decode_MPG2=0x00000000

  decode_WVC1=0x00000000

- **Vérification**

  vcgencmd codec_enabled MPG2

  vcgencmd codec_enabled WVC1

Config.txt

The Raspberry Pi uses a configuration file instead of the BIOS you would expect to find on a conventional PC. The system configuration parameters, which would traditionally be edited and stored using a BIOS, are stored instead in an optional text file named `config.txt` This is read by the GPU before the ARM CPU and Linux are initialised. It must therefore be located on the first (boot) partition of your SD card, alongside `bootcode.bin` and `start.elf`. This file is normally

Raspberry Pi boot sequence

- **Bootloader 1 (GPU) : boot ROM**

  Montage SD-Card

- **Bootloader 2 (GPU) : /boot/bootcode.bin**

  Drivers, boots du firmware GPU

- **GPU Firmware (GPU) : /boot/start.elf**

  Lecture de config.txt

- **Linux Kernel (CPU) : /boot/kernel.img**

GPU

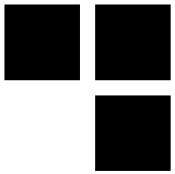
- **Broadcom VideoCore IV**
- **Documentation publique depuis 2014**

  Documentation PDF

  Headers C
- **Plugin IDA Pro**

  https://github.com/hermanhermitage/videocoreiv/tree/master/idaplugin

Reverse GPU

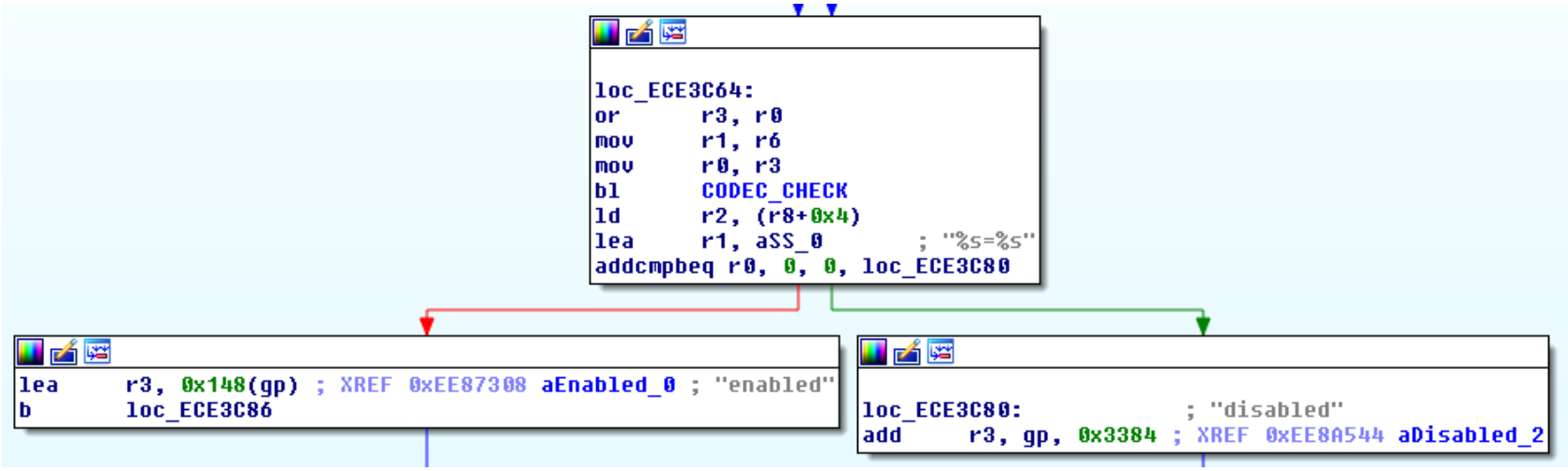


- **vcgencmd codec_enabled MPG2**

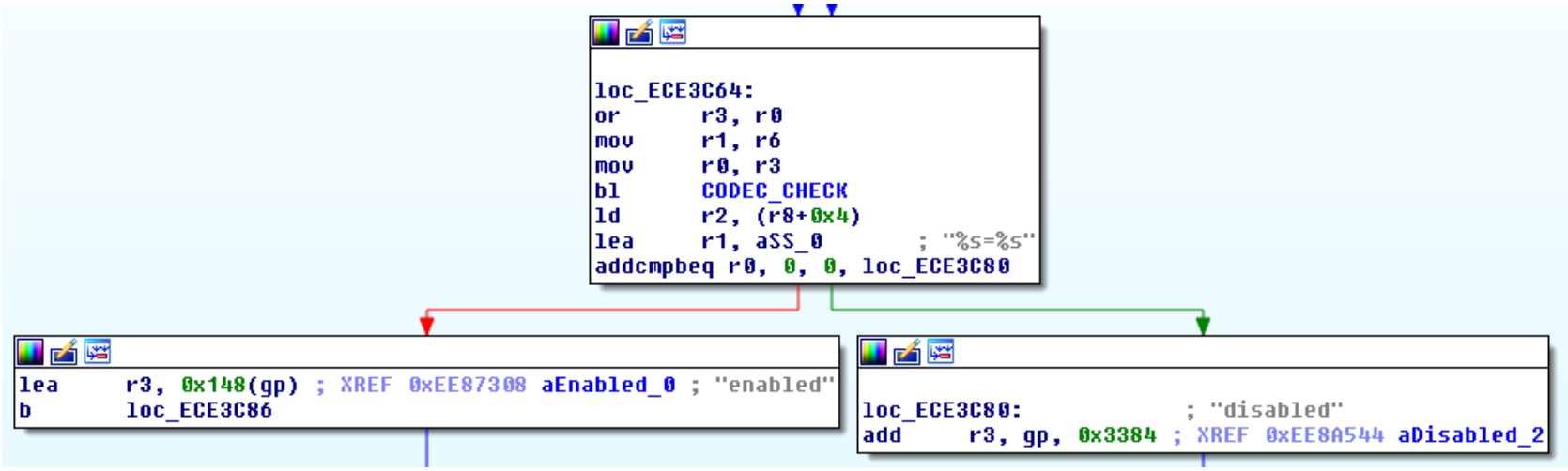
  vcgencmd : envoi de commandes au GPU

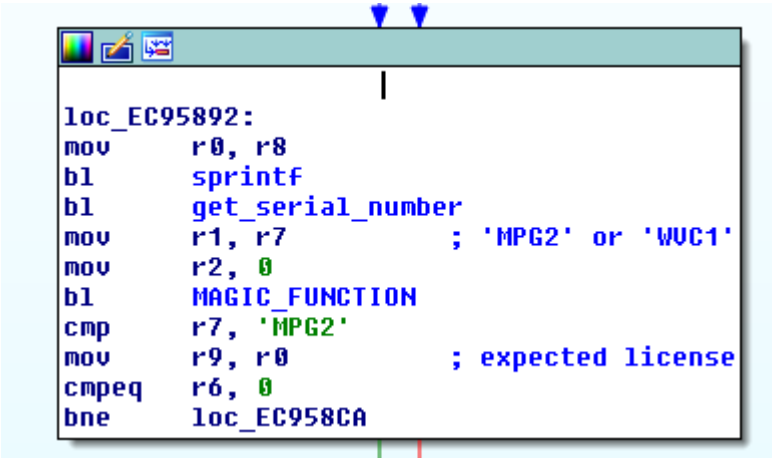
```
lea     r4, otp_dump

not     r0, 0
mov     r2, 1
mov     r3, 1
bl      register_cmd
lea     r1, 0x42C(pc)    ; "test_result"
lea     r4, sub_ECE63AE

not     r0, 0
mov     r2, 1
mov     r3, 1
bl      register_cmd
lea     r1, 0x4B4(pc)    ; "codec_enabled"
lea     r4, codec_enabled

not     r0, 0
mov     r2, 0x2
mov     r3, 0x2
bl      register_cmd
lea     r1, 0x3B0(pc)    ; "get_camera"
lea     r4, sub_ECE4378

not     r0, 0
mov     r2, 1
mov     r3, 1
bl      register_cmd
lea     r1, 0x354(pc)    ; "get_mem"
lea     r4, get_mem
```

Reverse GPU (2)



Reverse GPU (2)





Reverse GPU – WTFBBQ?

```
loc_EC9E23C:
bl      sub_EDF3410
mov     r9, r0
lea     r0, 0x2F(sp)
ld      r4, (r9+0xC)
lea     r1, 0x30(sp)
bl      r4
ld      r0, (sp+0x30)
mov     r1, 1
ld      r2, (r9+0x14)
bl      r2
ld      r0, (sp+0x30)
mov     r1, 0
ld      r3, (r9+0x3C)
mov     r2, 0
ld      r0, (sp+0x30)
add     r1, gp, 0xFFDA8480 ; XREF 0x10EC2F680 dword_EC2F680
ld      r3, (r9+0x3C)
add     r2, gp, 0xFFDA85AC ; XREF 0x10EC2F7AC dword_EC2F7AC
sub     r2, r1
bl      r3
mov     r1, 0
ld      r5, 0x40(r9)
mov     r2, 0
ld      r0, (sp+0x30)
add     r1, gp, 0xFFDA85AC ; XREF 0x10EC2F7AC dword_EC2F7AC
ld      r4, 0x40(r9)
add     r2, gp, 0xFFDA86AC ; XREF 0x10EC2F8AC dword_EC2F8AC
sub     r2, r1
bl      r4
mov     r1, 0x3
ld      r4, (r9+0x30)
mov     r2, 0
bl      r4
ld      r0, (sp+0x30)
mov     r1, 1
ld      r5, (r9+0x30)
mov     r2, r8
bl      r5
ld      r0, (sp+0x30)
addcmpbeq r6, 0, 0, loc_EC9E2B6
```

```
mov     r2, 0x137AFEDA
b       loc_EC9E2BC
```

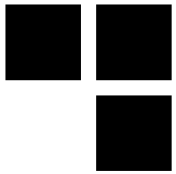
```
loc_EC9E2B6:
mov     r2, 0xF00BAD34
```

```
EF 0xEE88EB4 dword_EE88EB4
```

```
loc_EC9E2BC:
ld      r3, (r9+0x30)
mov     r1, 0x2
eor     r2, r7
mov     r10, sp
bl      r3
mov     r0, r10
lea     r1, unk_EE509D0
mov     r2, 0x2C
bl      sub_ED6886E
ld      r0, (sp+0x30)
not     r2, 0
ld      r5, (r9+0x20)
mov     r1, r10
shl     r2, 0x1E
bl      r5
ld      r0, (sp+0x30)
mov     r1, 0x2
ld      r3, (r9+0x34)
bl      r3
ld      r4, (r9+0x18)
mov     r10, r0
ld      r0, (sp+0x30)
bl      r4
ld      r0, (sp+0x30)
ld      r1, (r9+0x10)
st      r8, 0x1CB8(gp) ; XREF 0xEE88EB8 dword_EE88EB8
st      r7, 0x1CAC(gp) ; XREF 0xEE88EAC dword_EE88EAC
stb     r6, 0x1CA8(gp) ; XREF 0xEE88EA8 dword_EE88EA8
st      r10, 0x1CB4(gp) ; XREF 0xEE88EB4 dword_EE88EB4
```

```
loc_EC9E30A:
mov     r0, r10
lea     sp, 0x34(sp)
ldm     r6-r16, pc, (sp++)
; End of function sub_EC9E208
```

Such cheating, much study



- **Valeur de retour écrite en mémoire**

- **vcdbg grep**

  Usage: vcdbg grep WORD - Search memory for a 32-bit word

- **vcdbg dump**

  Usage: vcdbg dump ADDR|SYM [LEN] - Dumps memory (Hex and ASCII)

Censored

- **Censored**

Continuons l’etude

- **Ecriture de 2 blocs**

  0x12C bytes en 0x7F110000

  0x100 bytes en 0x7F100000

```
00000000: 546f 766b 94ce 1a57 5651 0cb2 72c9 c312  Tovk...WVQ..r...
00000010: 13bc e8d2 5ba3 2d2a 5a62 4deb 1640 0587  ....[.-*ZbM..@..
00000020: e098 39f7 acc6 ab7c e9fb 07aa 29cd 1d9b  ..9....|....)...
00000030: f60e 01bb 5cfc 15ae d9fa 9cef f175 8e70  ....\........u.p
00000040: 468b b089 50af 6e67 18da eed4 32be 4e58  F...P.ng....2.NX
00000050: 5d1f 4b73 88c0 7902 de47 a043 9adb c835  ].Ks..y..G.C...5
00000060: 953c cc8d 642f 1468 0071 03b9 ed0b f324  .<..d/.h.q.....$
00000070: 60b1 1763 df48 41a4 285e 2bd8 b490 ba83  `..c.HA.(^+.....
00000080: e408 d0e2 b86a 1074 9f7b 1938 8f91 d6a8  .....j.t.{.8....
```

I can see patterns...

```
00000000: 0110 1090 1f10 1480 8046 1004 a026 0804  .........F...&..
00000010: 0620 1090 1a20 1480 8046 1004 a016 0404  . ... ...F......
00000020: 0d10 1090 1310 1480 8046 1004 a026 0804  .........F...&..
00000030: 1120 1090 0f20 1480 8046 1004 a016 0404  . ... ...F......
00000040: 1510 1090 0b10 1480 8046 1004 a026 0804  .........F...&..
00000050: 1d20 1090 0320 1480 8046 1004 a016 0404  . ... ...F......
00000060: 1810 1090 ff40 10c4 0040 10b0 0040 1020  .....@...@...@.
00000070: 1840 1080 1010 1490 ff50 14c4 0050 14b0  .@.......P...P..
00000080: 0050 1420 1050 1480 8046 1004 0810 1490  .P. .P...F......
00000090: ff50 14c4 0050 14b0 0050 1420 0850 1480  .P...P...P. .P..
000000a0: 8046 1004 ff10 14c4 0050 14b0 0050 1420  .F.......P...P.
```

Potentiellement du code pour un CPU RISC 32b

Memory addresses

hardware_vc4.h

```
#define VCE_BASE 0x7f100000
#define VCE_DATA_MEM_OFFSET 0
#define VCE_DATA_MEM_SIZE 0x2000
#define VCE_PROGRAM_MEM_OFFSET 0x10000
#define VCE_PROGRAM_MEM_SIZE 0x4000
#define VCE_REGISTERS_OFFSET 0x20000
#define VCE_REGISTERS_COUNT 63
#define VCE_STATUS_OFFSET 0x40000
```

Documentation

- **Aucune référence à VCE dans la documentation**

  On ne trouve que quelques drivers Android

- **Obscure référence dans un brevet Broadcom**

  The hardware video accelerator 216 may comprise a plurality of components and/or modules, such as a ***video control engine (VCE)***

Quelques drivers

- **android-bcm-tetra-3.10-kitkat-wear**

  /drivers/char/broadcom/vce/*

  /drivers/char/broadcom/mm/h264/vce.c

- **Permet de reconnaître certaines fonctions**

Reverse GPU – Now with comments!

```
bl      get_vce_driver
mov     r9, r0              ; vce_driver
lea     r0, 0x2F(sp)
ld      r4, (r9+0xC)
lea     r1, 0x30(sp)
bl      r4                  ; vce_open
ld      r0, (sp+0x30)
mov     r1, 1
ld      r2, (r9+0x14)
bl      r2                  ; vce_obtain_semaphore
ld      r0, (sp+0x30)
mov     r1, 0
ld      r3, (r9+0x3C)
mov     r2, 0
bl      r3                  ; vce_loadprogram
ld      r0, (sp+0x30)
add     r1, gp, 0xFFDA84C0 ; XREF 0x10EC2F680 byte_EC2F680
ld      r3, (r9+0x3C)
add     r2, gp, 0xFFDA85EC ; XREF 0x10EC2F7AC byte_EC2F7AC
sub     r2, r1
bl      r3                  ; vce_loadprogram
ld      r0, (sp+0x30)
mov     r1, 0
ld      r5, 0x40(r9)
mov     r2, 0
bl      r5                  ; vce_loaddata
ld      r0, (sp+0x30)
add     r1, gp, 0xFFDA85EC ; XREF 0x10EC2F7AC byte_EC2F7AC
ld      r4, 0x40(r9)
add     r2, gp, 0xFFDA86EC ; XREF 0x10EC2F8AC loc_EC2F8AC
sub     r2, r1
bl      r4                  ; vce_loaddata
ld      r0, (sp+0x30)
mov     r1, 0x3
ld      r4, (r9+0x30)
mov     r2, 0
bl      r4                  ; vce_setreg
ld      r0, (sp+0x30)
mov     r1, 1
ld      r5, (r9+0x30)
mov     r2, r8
bl      r5                  ; vce_setreg
ld      r0, (sp+0x30)
addcmpbeq r6, 0, 0, loc_EC9E296
```

```
mov     r2, 0x137AFEDA
b       loc_EC9E29C
```

```
loc_EC9E296:
mov     r2, 0xF00BAD34
```

```
loc_EC9E29C:
ld      r3, (r9+0x30)
mov     r1, 0x2
eor     r2, r7
mov     r10, sp
bl      r3                  ; vce_setreg
mov     r0, r10
lea     r1, off_EE50990
mov     r2, 0x2C
bl      memcpy
ld      r0, (sp+0x30)
not     r2, 0
ld      r5, (r9+0x20)
mov     r1, r10
shl     r2, 0x1E
bl      r5                  ; sub_EDF3728 pony pony run run
ld      r0, (sp+0x30)
mov     r1, 0x2
ld      r3, (r9+0x34)
bl      r3                  ; vce_getreg
ld      r4, (r9+0x18)
mov     r10, r0
ld      r0, (sp+0x30)
bl      r4                  ; vce_release_semaphore
ld      r0, (sp+0x30)
ld      r1, (r9+0x10)
bl      r1                  ; vce_close
st      r8, 0x1CB8(gp) ; XREF 0xEE88E78 dword_EE88E78
st      r7, 0x1CAC(gp) ; XREF 0xEE88E6C dword_EE88E6C
stb     r6, 0x1CA8(gp) ; XREF 0xEE88E68 dword_EE88E68
st      r10, 0x1CB4(gp) ; XREF 0xEE88E74 dword_EE88E74
```

Instrument all the things

Instrument all the things (2)

- **Modification de start.elf**

  On peut lire les registres du VCE

  On peut choisir le code que l'on charge

- **Exécutons les instructions pas à pas !**

  Les registres nous donneront l'instruction exécutée

Start

PC	0x00000000
R0	0x00000000
R1	0xb7380b72
R2	0xb04bed74
R3	0x00000000
R4	0x????????
R5	0x????????

Start + 4

PC	**0x00000004**
R0	0x00000000
R1	0xb7380b72
R2	0xb04bed74
R3	0x00000000
R4	**0x5b9c05b9**
R5	0x????????

Instruction
90 10 10 01
1001 0000 0001 0000 0001 0000 0000 0001

Start + 4

PC	**0x00000004**
R0	0x00000000
R1	0xb7380b72
R2	0xb04bed74
R3	0x00000000
R4	**0x5b9c05b9**
R5	0x????????

Instruction
90 10 10 01
1001 0000 0001 0000 0001 0000 0000 0001

R4 = SHR(R1 , 1)

Start + 4

PC	**0x00000004**
R0	0x00000000
R1	0xb7380b72
R2	0xb04bed74
R3	0x00000000
R4	**0x5b9c05b9**
R5	0x????????

Instruction
90 10 1**0 01**
1001 0000 0001 0000 0001 0000 0000 0001

R4 = SHR(R1 , **1**)

Start + 4

PC	**0x00000004**
R0	0x00000000
R1	0xb7380b72
R2	0xb04bed74
R3	0x00000000
R4	**0x5b9c05b9**
R5	0x????????

Instruction
90 10 1**0 01**
1001 0000 0001 0000 0001 0000 0000 0001

R4 = SHR(**R1** , **1**)

Start + 4

PC	**0x00000004**
R0	0x00000000
R1	0xb7380b72
R2	0xb04bed74
R3	0x00000000
R4	**0x5b9c05b9**
R5	0x????????

Instruction
90 10 1**0 01**
1001 0000 0001 0000 0001 0000 0000 0001

R4 = SHR(**R1** , **1**)

Start + 4

PC	**0x00000004**
R0	0x00000000
R1	0xb7380b72
R2	0xb04bed74
R3	0x00000000
R4	**0x5b9c05b9**
R5	0x????????

Instruction
90 10 1**0 01**
1001 0000 0001 0000 0001 0000 0000 0001

R4 = **SHR**(**R1** , **1**)

Start + 8

PC	**0x00000008**
R0	0x00000000
R1	0xb7380b72
R2	0xb04bed74
R3	0x00000000
R4	0x5b9c05b9
R5	**0x00000000**

Instruction
80 14 10 1F
1000 0000 0001 0100 0001 0000 0001 1111

R5 = ???

Start + 8

PC	**0x00000008**
R0	0x00000000
R1	0xb7380b72
R2	0xb04bed74
R3	0x00000000
R4	0x5b9c05b9
R5	**0x00000000**

Instruction
80 14 10 1F
1000 0000 0001 0100 0001 0000 0001 1111

R5 = ???

Start + 8

PC	**0x00000008**
R0	0x00000000
R1	0xb7380b72
R2	0xb04bed74
R3	0x00000000
R4	0x5b9c05b9
R5	**0x00000000**

Instruction
80 14 10 1F
1000 0000 0001 0100 0001 0000 0001 1111

R5 = **R1** ???

Start + 8

PC	**0x00000008**
R0	0x00000000
R1	0xb7380b72
R2	0xb04bed74
R3	0x00000000
R4	0x5b9c05b9
R5	**0x00000000**

Instruction
80 14 1**0 1F**
1000 0000 0001 0100 0001 0000 0001 1111

R5 = **R1** ??? **0x1f**

Start + 8

PC	**0x00000008**
R0	0x00000000
R1	0xb7380b72
R2	0xb04bed74
R3	0x00000000
R4	0x5b9c05b9
R5	**0x00000000**

Instruction
80 14 1**0** **1F**
1000 0000 0001 0100 0001 0000 0001 1111

R5 = **SHL**(**R1** , **0x1f**)

Start + C

PC	**0x0000000C**
R0	0x00000000
R1	0xb7380b72
R2	0xb04bed74
R3	0x00000000
R4	0x5b9c05b9
R5	0x00000000

Instruction
04 10 46 80
0000 0100 0001 0000 0100 0110 1000 0000

R??? = ???

Start + C

PC	**0x0000000C**
R0	0x00000000
R1	0xb7380b72
R2	0xb04bed74
R3	0x00000000
R4	0x5b9c05b9
R5	0x00000000

Instruction
04 10 46 80
0000 0100 0001 0000 0100 0110 1000 0000

R4 = ???

Start + 10

PC	**0x00000010**
R0	0x00000000
R1	0xb7380b72
R2	**0xebd7e8cd**
R3	0x00000000
R4	0x5b9c05b9
R5	0x00000000

Instruction
04 08 26 a0
0000 0100 0000 1000 0010 0110 1010 0000

R2 = **R2** **XOR** R4

Procédure

- **Exécution des instructions pas à pas**
- **Recherche des liens entre les registres**
- **Déduction des instructions**
- **Ecriture d’un désassembleur**

Instruction set

0x90	SHR
0x80	SHL
0x04	XOR
0xC4	AND
0x20	LD
0xB0	ADD
0xE8	MOV
0xF0	CMP
0x7?	JCC

Censored

- **Censored**

C’est fini

Avez-vous des questions ?

On peut boire les bières ?

Quelqu’un a parlé de rhum ?