

# CHIP RED PILL: КАК НАМ УДАЛОСЬ ВЫПОЛНИТЬ ПРОИЗВОЛЬНЫЙ [МИКРО]КОД ВНУТРИ ПРОЦЕССОРОВ INTEL ATOM

 ***\_Dmit***      ***Dmitry Sklyarov***  
 ***h0t\_max***      ***Maxim Goryachy***  
 ***\_markel\_\_***      ***Mark Ermolov***

# Исследовательская группа



Mark Ermolov

- Positive Technologies
  - Lead Expert
- mermolov@ptsecurity.com*



Dmitry Sklyarov

- Head of Reverse Engineering
  - Positive Technologies
- dsklyarov@ptsecurity.com*



Maxim Goryachy

- Firmware/Hardware bug hunter
  - ex-Positive Technologies
  - Независимый исследователь
- h0t\_max@hotmail.com*

# Содержание

- Что такое микрокод
- Доступ к внутренним шинам процессоров Intel
- Реверс-инжиниринг микрокода процессора Intel
- Расшифровка microcode update

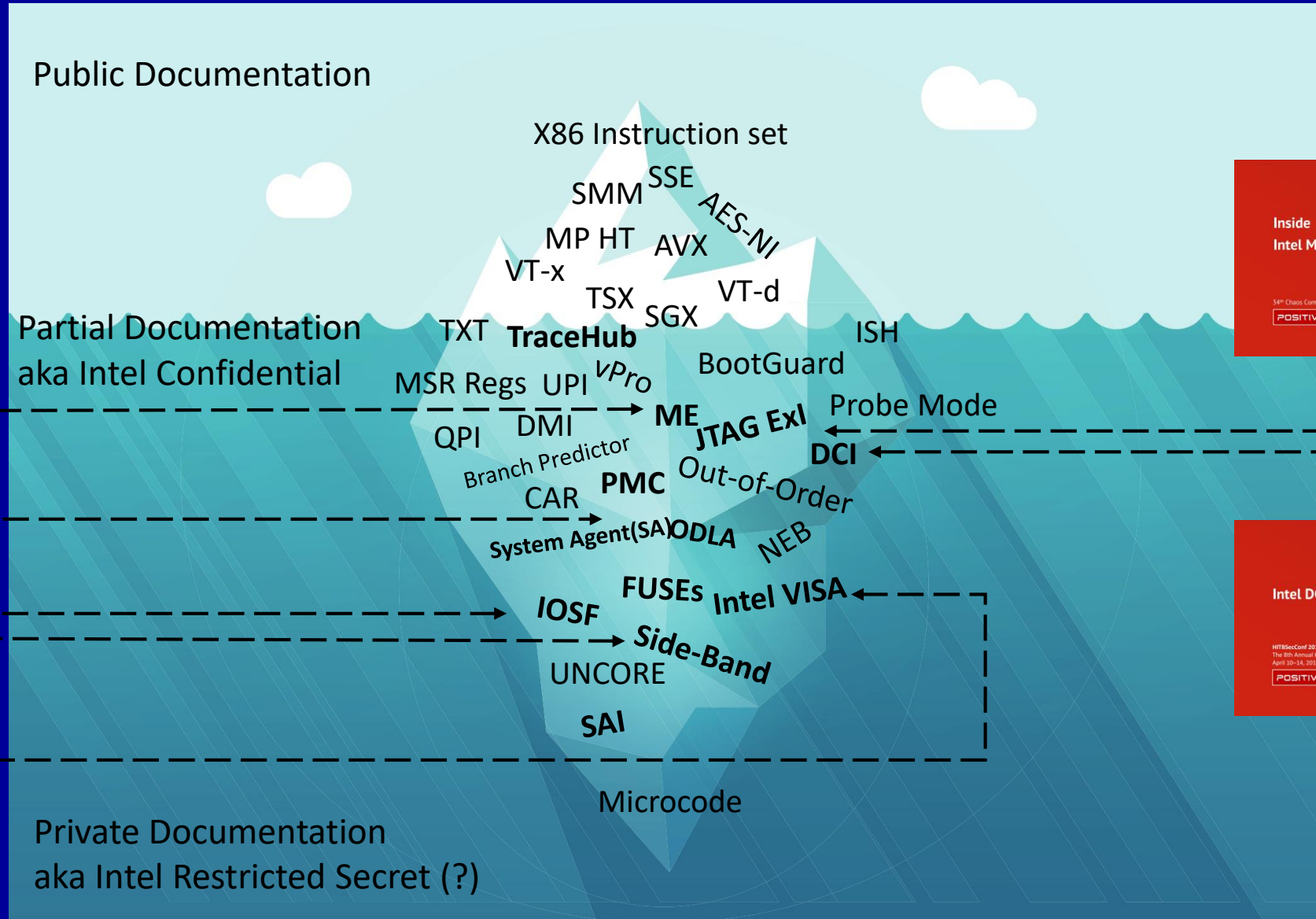
# Содержание

- Что такое микрокод
- Доступ к внутренним шинам процессоров Intel
- Реверс-инжиниринг микрокода процессора Intel
- Расшифровка microcode update

# Технологии процессоров Intel



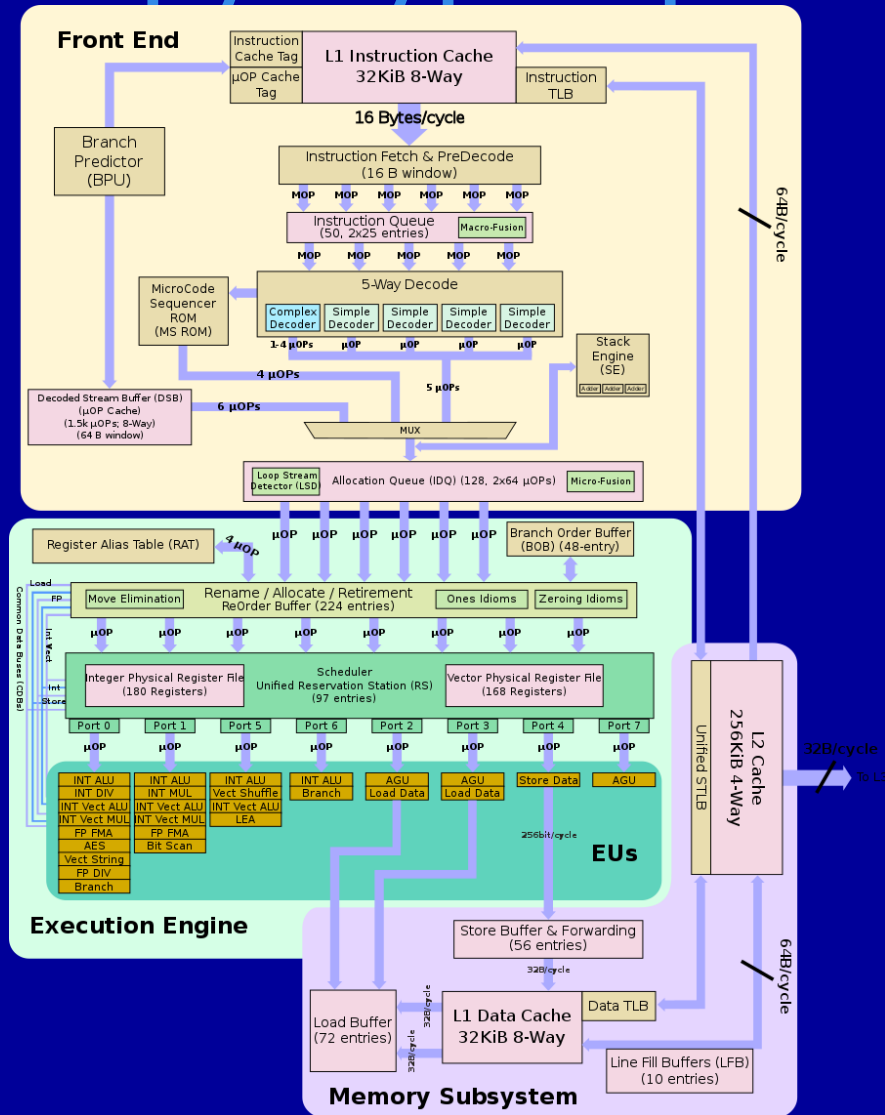
# Наши предыдущие исследования платформ Intel



# Что такое микрокод?

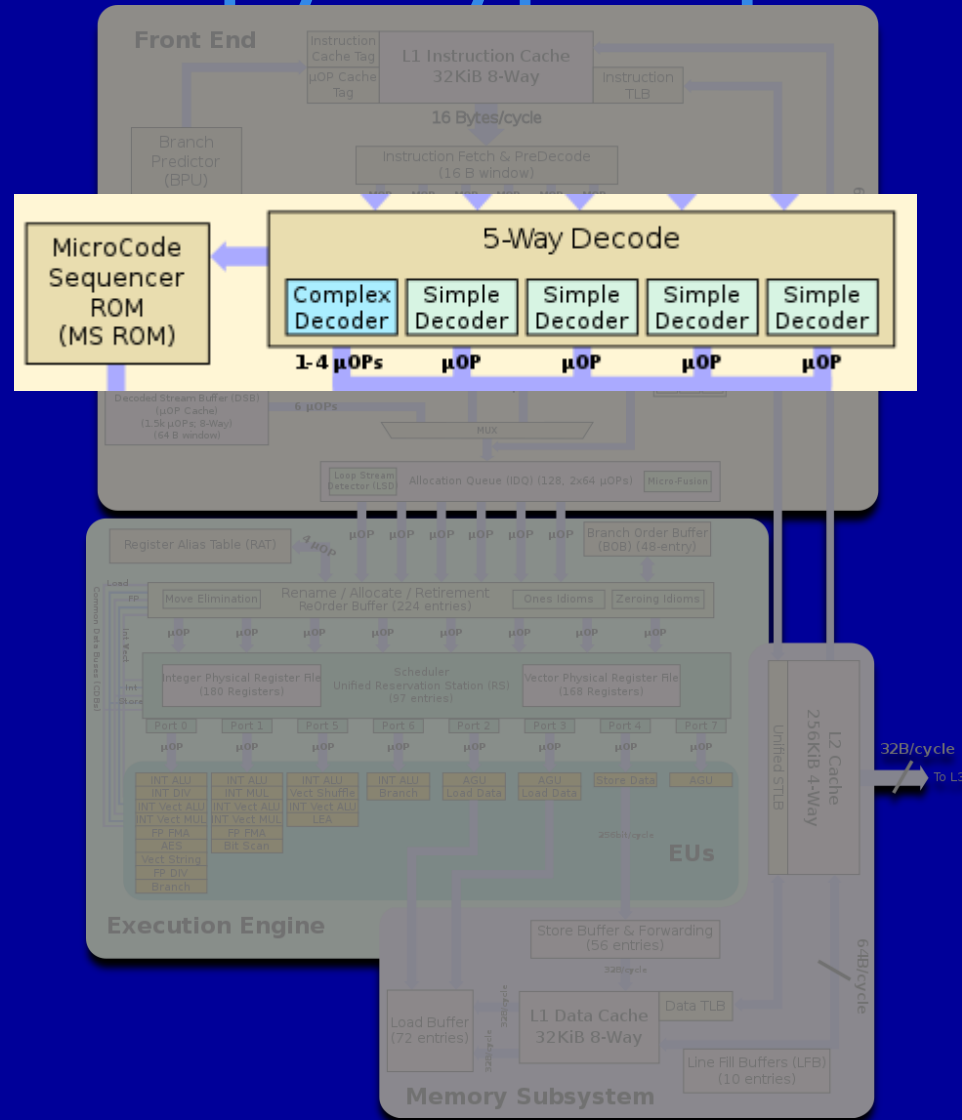
- Инициализационный код ЦПУ;
- Реализует некоторые технологии безопасности (???: SGX, VT-x, MPX, TXT);
- Управляет энергопотреблением;
- Имеет возможность установки обновлений;
- Архитектурно зависим.

# Внутренняя структура ЦПУ Intel

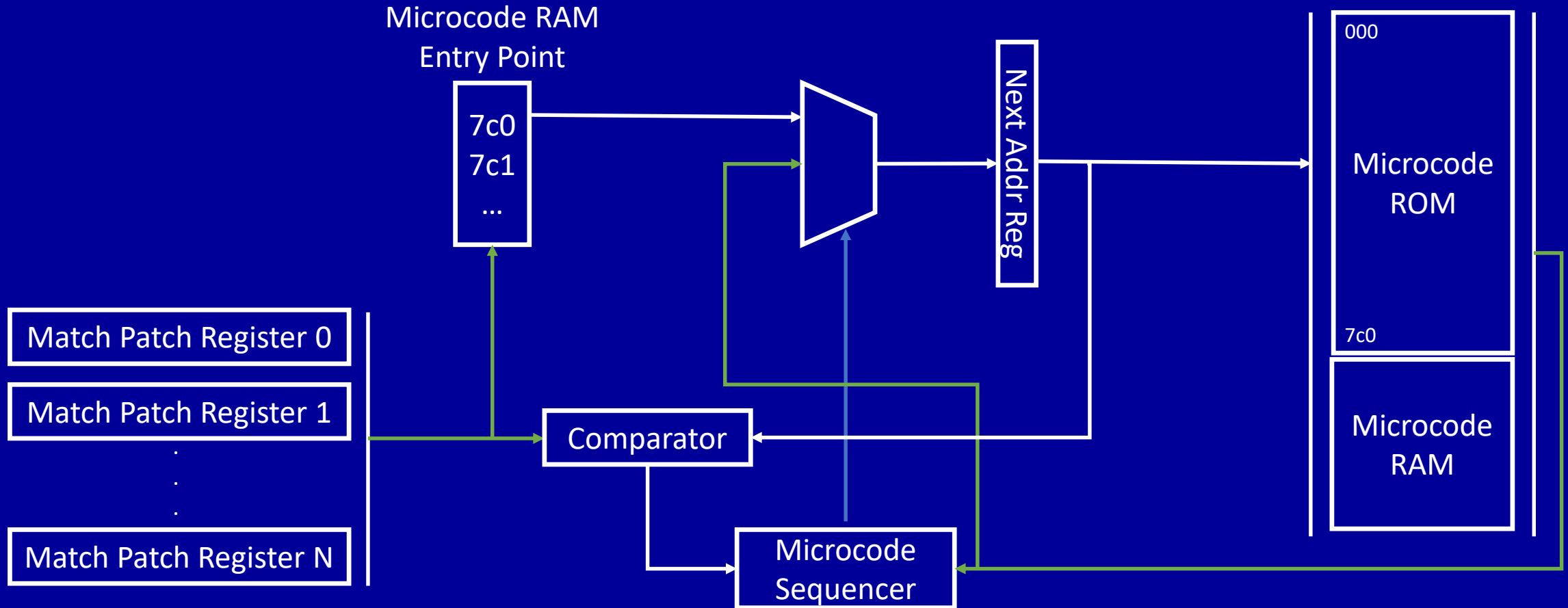




# Внутренняя структура ЦПУ Intel



# Устройство декодера uCode

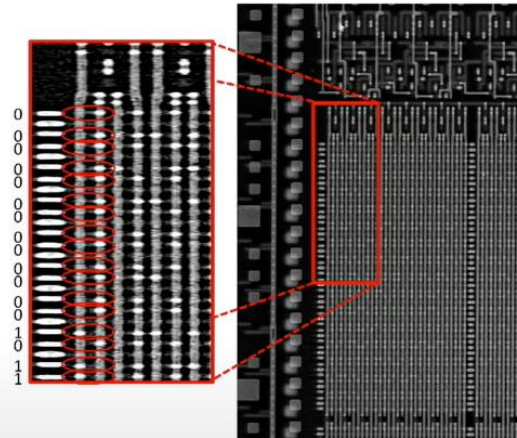


# Содержание

- Что такое микрокод
- Доступ к внутренним шинам процессоров Intel
- Реверс-инжиниринг микрокода процессора Intel
- Расшифровка microcode update

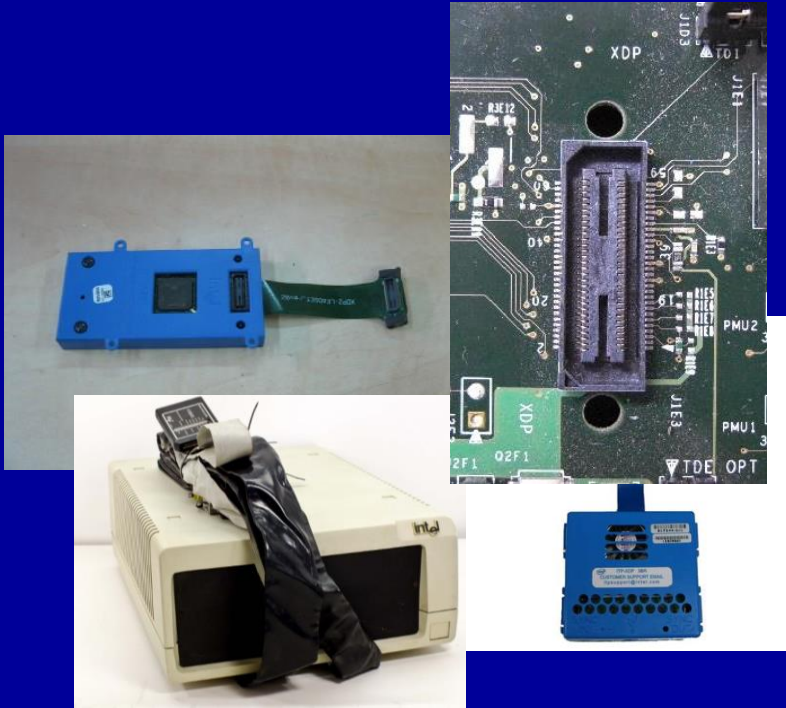
# Подход к изучению: Аппаратный

## Analysis - Hardware



*“Reverse Engineering x86 Processor Microcode”  
Philipp Koppe, Benjamin Kollenda, Marc Fyrbiak, Christian  
Kison, Robert Gawlik, Christof Paar, and Thorsten Holz*

# Подход к изучению: Программный



Специализированные материнские платы



USB, пользовательское оборудование

# Intel PCH JTAG Unlock

- Мы смогли активировать инженерную отладку для микросхемы PCH;
- Полный доступ к Intel Management Engine;
- Intel ME не имеет микрокода до поколения Ice Lake (2020+);



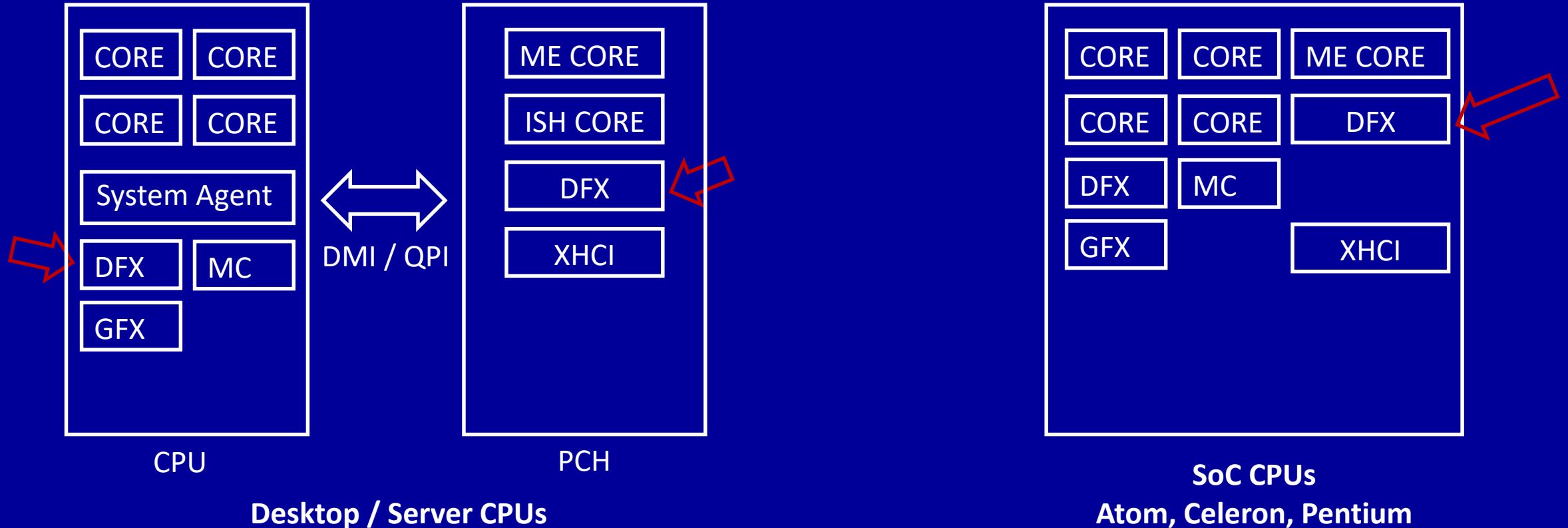
# Пароль для JTAG Unlock

```
<_tdef deviceType="SKL_M_UC" tables="TapDefs">
  <_tdefDevice steppings="M0">
    <TapDefs>
      <TapCommands>
        <TapCommand Ir="0x042" TapRegister="SA_TAP_LR_GLOBALUNLOCK" _name="SA_TAP_LR_GLOBALUNLOCK" Register="READWRITE" />
        <TapCommand Ir="0x048" TapRegister="SA_TAP_LR_REDUNLOCK" _name="SA_TAP_LR_REDUNLOCK" Register="READWRITE"
PreScan="PackageAndC10Wakeup.Awake" />
        <TapCommand Ir="0x049" TapRegister="SA_TAP_LR_ORANGEUNLOCK" _name="SA_TAP_LR_ORANGEUNLOCK" Register="READWRITE"
PreScan="PackageAndC10Wakeup.Awake" />
        <TapCommand Ir="0x04A" TapRegister="SA_TAP_LR_UNIQUEID" _name="SA_TAP_LR_UNIQUEID" Register="READ1" PreScan="PackageAndC10Wakeup.Awake" />
      </TapCommands>
      ...
      <TapRegister _indices="71:0" _map="" _name="SA_TAP_LR_REDUNLOCK">
        <_tag key="Visible" value="false" />
        <Field _map="71:0" _name="DR_PASSWORD" />
      </TapRegister>
    </_tdefDevice>
  </_tdef>

```

```
[CE4200]
datfile=jcs\ia7muxia7.jcs
xdp3jtagtclk=10000000
mcrjtagtclk=4000000
cpu#0=1,11,tci-jtag-gen-ia.dll,CE4200,HT0,Master
cpu#1=2,11,tci-jtag-gen-ia.dll,CE4200,HT1,Slave
...
STUB=jtag.ini
passwd01=13,64,31
passwd02=64,101,77,107,67,111,76,110,85 ;65 4D 6B 43 6F 4C 6E 55
passwd03=13,2,0
passwd04=16,2,2 ; make this 7,3 to enable the 8051 on secondary chain
```

# Связь между Intel ME and CPU



Если SoC имеет только одно DFX устройство, значит ли это, что разблокировка ME дает разблокировку ЦПУ?



# Извлечение Intel Atom Microcode



Mark Ermolov  
@\_markel\_\_

Finally, the casket is opened: we (+@h0t\_max and @\_Dmit) have extracted Intel x86 microcode! One more Intel "top secret" information gets revealed...  
[github.com/chip-red-pill/...](https://github.com/chip-red-pill/)

```
nd Prompt - python
rgs_dump()
00003e573a3b 00003e8ff6ef7 00003e8c6217
00003e5d69ef 00003e1b18b3 00003e1f2833
00003e2f23ab 00003e042011 00003e0018dd
00003e854c33 00003e553a03 00003e533603
00003e77738f 000000000000 000000000000
000000000000 000000000000 000000000000
000000000000 000000000000 000000000000
000000000000 000000000000 000000000000
000000000000 000000000000 000000000000
*)
015d757002c0 815d757002c0 41510000fb0
:062f01f1200 404337000235 417000035d71
114208e00f80 00012b039e48 00002003cf08
04800035472 80070043ef9f 400305031c80
190205c00200 811f0003f03f 8158047002c0
003005030c00 9062010f2240 c00524071e00
0050003dc7f 40150003f23f 40e100030032
596206c00240 03000003f03e 00040303ffc8
0054703ffc8 40620103f200 c0a40503e23e
:0410003efbf c0637f03f200 00520c036200
e86a446d023f c06350032200 00400403ef00
786a11310631 406387030200 b86ab03102f1
cd8bc443f00a c0010003ffffe 40070103ffc8
00070103ffc8 0e750003003c 800610131e08
004a1032c90 803200032cb0 7929e42c0032
1131010b1231 0001000311c7d 700f00035c88
:00500078e00 00000103d000 c0330003bd7b
:007fc035d40 800a20000200 c150197402fb
003200032cb0 7929e42c0032 806353030200
00360003cf38 e30000030c00 c0a100031ef1
```

1:52 PM - May 19, 2020 - Twitter Web Client

# Agenda

- Microcode Overview
- Access to CPU's internals
- Intel microcode reversing
- Decrypting microcode update

# LDATs

ARRAY0

```
0000: 00626803f200 000801030008 004800013000 000000000000
0004: 05b900013000 000a01000200 014800000000 000000000000
...
7ffc: 000000000000 000000000000 000000000000 000000000000
```

RO

ARRAY1

```
0000: 0000018e5e40 0000018e5e40 0000018e5e40 0000018e5e40
0004: 00000b000240 00000b000240 00000b000240 00000b000240
...
7ffc: 0000018000c0 0000018000c0 0000018000c0 0000018000c0
```

RO

ARRAY2

```
0000: 0000070000ce 000018201a50 000018201a50 0000384c0600
...
007c: 000000000000 000000000000 000000000000 000000000000
```

RW

ARRAY3

```
0000: 000000000000 00003e5f3a3b 00003e996ef7 00003e966217
...
001c: 000000000000 000000000000 000000000000 000000000000
```

RW

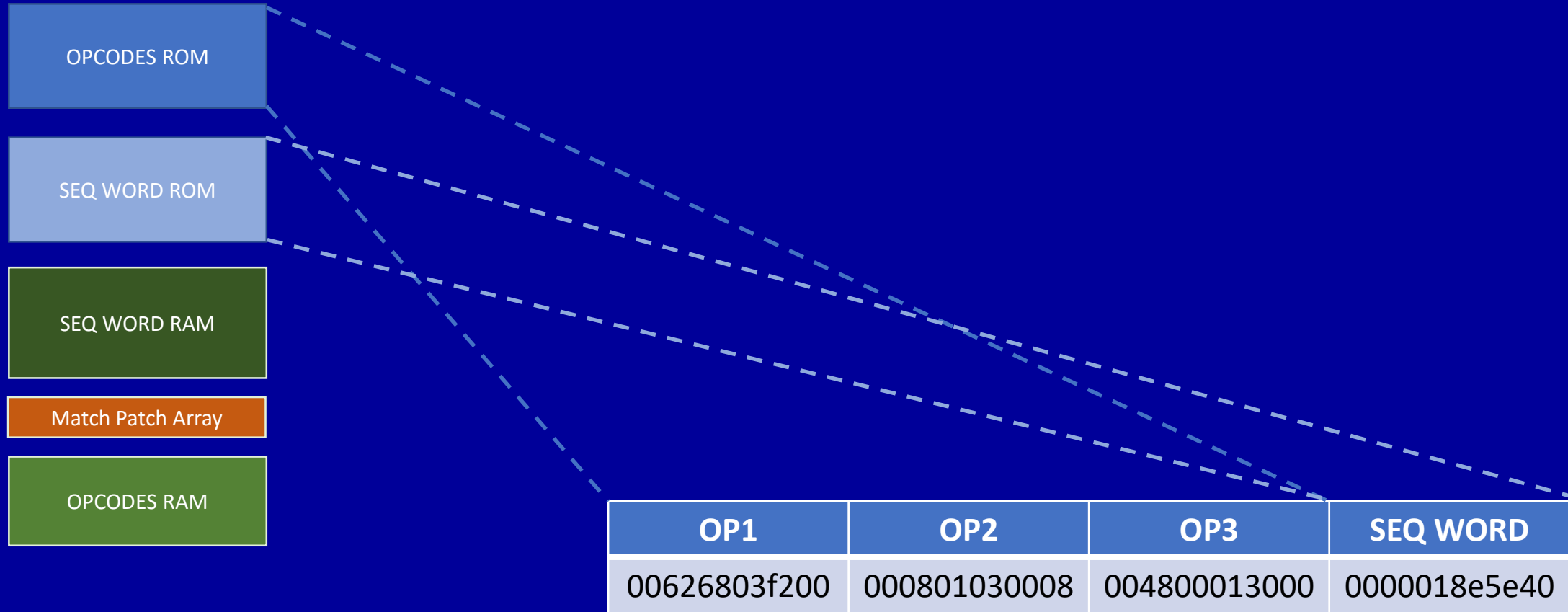
ARRAY4

```
0000: c0053d03ffc8 815d857002c0 815d857002c0 415100000fb0
...
01fc: 000000000000 000000000000 000000000000 000000000000
```

RW

[https://github.com/chip-red-pill/crbus\\_scripts](https://github.com/chip-red-pill/crbus_scripts)

# Связь между LDAT и uCode



# Формат uOps

	47..46	45	44	43..32	31..24	23	22..18	17..12	11..6	5..0
Field Name	PARITY	M1	M2	OPCODE	IMM0	M0	IMM1	DST	SRC1	SRC0
Size	2	1	1	12	8	1	5	6	6	6

**OPCODE** - 12-bit numeric microoperation code of operation

**SRC0/SRC1/DST** - three 6-bits fields which select operands for the operation

**M0/M1/M2** - bits representing modes of the operation

**IMM0/IMM1** – represent bits #0-7 and #8-12 of immediate values embedded directly into uops.

# Формат Sequence Word

	29..28	27..25	24..23	22..8	7..6	5..2	1..0
Field Name	PARITY	SYNC	UP2	UADDR	UP1	EFLOW	UP0
Size	2	3	2	15	2	4	2

**UP0/UP1/UP2** – 2-bit pointers to microoperation inside triad.

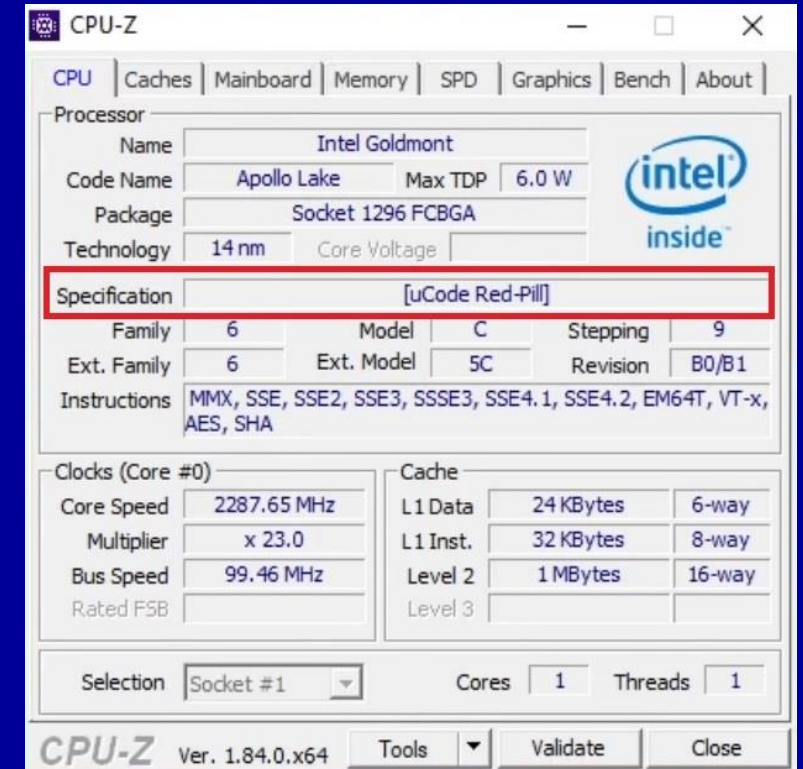
**EFLOW** – 4-bit field that controls execution flow for the microoperations triad.

**UADDR** – 15-bit field that specifies the address in microcode ROM/RAM

**SYNC** – 3-bit field that controls two synchronization aspects those apply for microoperations execution

# Выполнение произвольного uCode

- Обнаружили точку входа CPUID[0x8000002.. 0x8000004]
- Разработали полезную нагрузку
- Провели разблокировку RED unlock
- Загрузили полезную нагрузку в LDAT array[2..4]



The screenshot shows the CPU-Z application window. The 'Processor' tab is selected, displaying the following information:

Name	Intel Goldmont		
Code Name	Apollo Lake	Max TDP	6.0 W
Package	Socket 1296 FCBGA		
Technology	14 nm	Core Voltage	
Specification	[uCode Red-Pill]		
Family	6	Model	C
Ext. Family	6	Ext. Model	5C
Stepping	9	Revision	B0/B1
Instructions	MMX, SSE, SSE2, SSE3, SSSE3, SSE4.1, SSE4.2, EM64T, VT-x, AES, SHA		

Below the processor information, there are sections for 'Clocks (Core #0)' and 'Cache'.

Core Speed	2287.65 MHz
Multiplier	x 23.0
Bus Speed	99.46 MHz
Rated FSB	

L1 Data	24 KBytes	6-way
L1 Inst.	32 KBytes	8-way
Level 2	1 MBytes	16-way
Level 3		

At the bottom, the 'Selection' dropdown is set to 'Socket #1', with 'Cores' set to 1 and 'Threads' set to 1. The CPU-Z version is 1.84.0.x64.

# Intel Atom uCode дизассемблер

```
U0000: 00626803f200      tmp15:= MOVEFROMCREG_DSZ64(CORE_CR_CUR_UIP)
U0001: 000801030008      tmp0:= ZEROEXT_DSZ32(0x00000001)
        018e5e40      SEQW GOTO U0e5e
-----
U0002: 004800013000      tmp7:= ZEROEXT_DSZ64(0x00000000)
U0004: 05b900013000      mm7:= unk_5b9(0x00000000)
U0005: 000a01000200      TESTUSTATE(UCODE, UST_MSLOOPCTR_NONZERO)
        0b000240      ? SEQW GOTO U0002
U0006: 014800000000      SYNCWAIT-> URET(0x00)
-----
```

<https://github.com/chip-red-pill>



# Agenda

- Microcode Overview
- Access to CPU's internals
- Intel microcode reversing
- Decrypting microcode update

# Что нам было известно о Microcode Update

- Микрокод ЦПУ обновляется из UEFI/OS;
- Зашифрован на неизвестном алгоритме;
- Имеет подпись, основанную на алгоритме RSA-2048.

# Формат Microcode Update

**Table 9-8. Microcode Update Format**

31	24	16	8	0	Bytes			
Header Version					0			
Update Revision					4			
Month: 8		Day: 8		Year: 16	8			
Processor Signature (CPUID)					12			
Res: 4	Extended Family: 8		Extended Mode: 4	Reserved: 2	Type: 2	Family: 4	Model: 4	Stepping: 4
Checksum					16			
Loader Revision					20			
Processor Flags					24			
Reserved (24 bits)					P0 P1 P2 P3 P4 P5 P6 P7			
Data Size					28			
Total Size					32			
Reserved (12 Bytes)					36			
Update Data (Data Size bytes, or 2000 Bytes if Data Size = 00000000H)					48			
Extended Signature Count 'n'					Data Size + 48			
Extended Processor Signature Table Checksum					Data Size + 52			
Reserved (12 Bytes)					Data Size + 56			
Processor Signature[n]					Data Size + 68 + (n * 12)			
Processor Flags[n]					Data Size + 72 + (n * 12)			
Checksum[n]					Data Size + 76 + (n * 12)			

# Расшифровка Atom uCode Update: Алгоритм

5ed5:

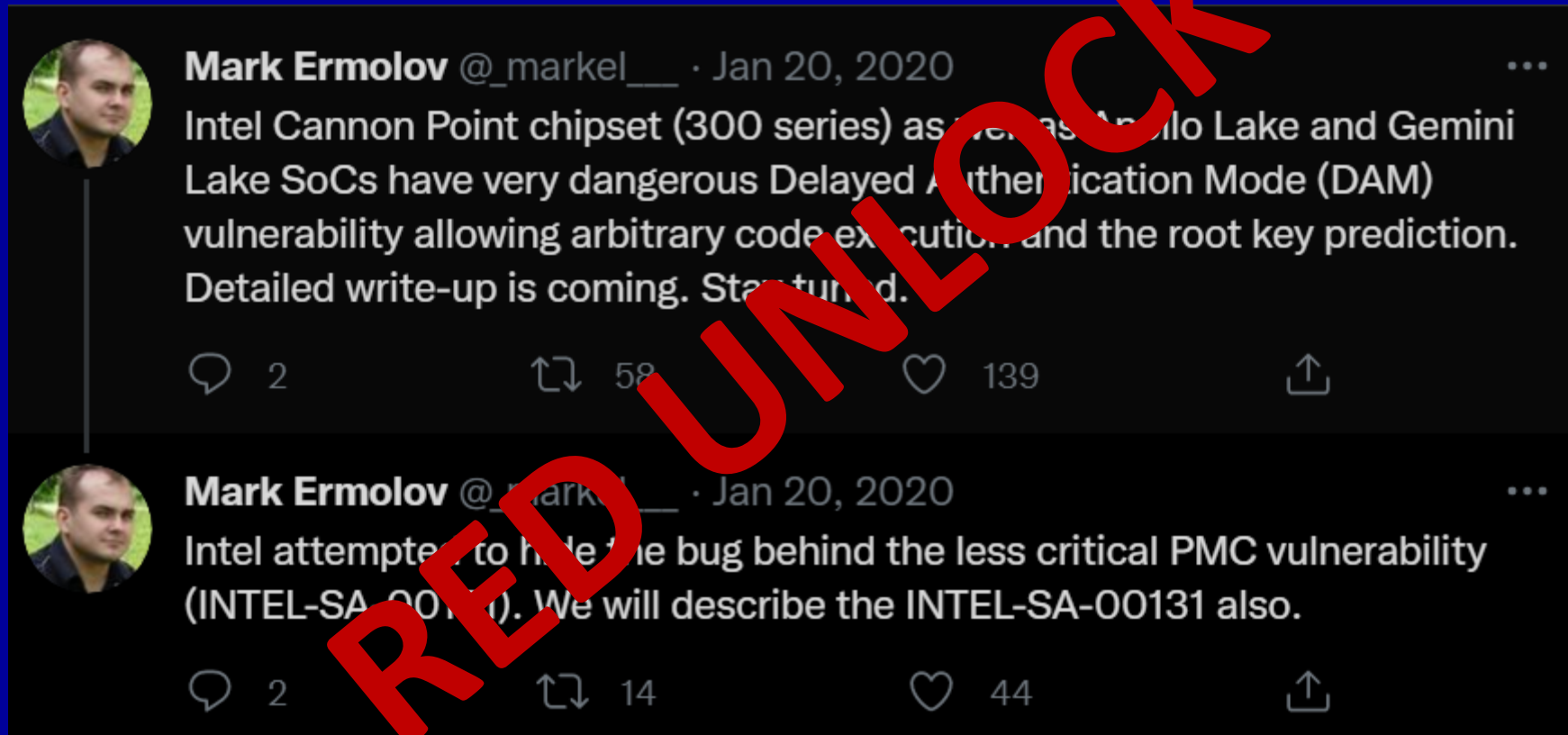
```
tmp0:= ADD_DSZ8(0x1, tmp0)
tmp2:= LDPPHYS_DSZ8_ASZ64_SC1(tmp7 + tmp0)
tmp1:= ADD_DSZ8(tmp2, tmp1)
tmp3:= LDPPHYS_DSZ8_ASZ64_SC1(tmp7 + tmp1)
STAPPHYS_DSZ8_ASZ64_SC1(tmp7 + tmp0, tmp3)
STAPPHYS_DSZ8_ASZ64_SC1(tmp7 + tmp1, tmp2)
tmp2:= ADD_DSZ8(tmp3, tmp2)
tmp2:= LDPPHYS_DSZ8_ASZ64_SC1(tmp7 + tmp2)
tmp3:= LDPPHYS_DSZ8_ASZ64_SC1(tmp5)
tmp3:= XOR_DSZ8(tmp2, tmp3)
STAPPHYS_DSZ8_ASZ64_SC1(tmp5, tmp3)
tmp5:= ADD_DSZ64(0x1, tmp5)
tmp6:= SUB_DSZ32(0x1, tmp6)
UJMPCC_DIRECT_CONDZ(tmp6, tmp8)
SEQWORD GOTO 0x5ed5

# i := (i + 1) mod 256
# S[i]
# j := (j + S[i]) mod 256
# swap values of S[i] and S[j]
#
# (S[i] + S[j]) mod 256
# K := S[(S[i] + S[j]) mod 256]
# *pb
#
# *pb ^= K
# pb++
# cb--
# if 0 == cb: GOTO tmp8
```

# Результат расшифровки Atom uCode Update

```
cpu506C9_plat0▶ ↓FR0 ----- 00000100 Hiew 8.68 (c)SEN
00000: 01 02 00 7C-39 00 0A 00-3F 88 4B ED-C0 00 08 0C 00 |9  ?ИКэ L ♀
00010: 0B 01 47 80-00 00 0A 00-3F 88 4F AD-00 03 0A 00 ♂@GA  ?ИОН ♥
00020: 2F 20 4B 2D-80 02 08 0C-03 22 47 40-A9 03 0A 00 / K-A@♀♥"G@й♥
00030: 2F 20 4F 6D-19 02 00 02-03 53 63 80-C0 00 30 02 / Om↓  ♥ScA L 00
00040: B8 A6 6B E8-00 00 00 02-03 20 63 C0-00 03 F0 03 җжкш  ♥ с L ♥Е♥
00050: F8 A6 6B 28-C0 00 08 00-03 C0 0B ED-00 00 0B 10 °жк( L ♥ L♂э ♂▶
00060: 7F 00 08 00-80 01 31 10-03 00 A1 40-C0 00 31 0C ♂ ♀ A@1▶♥ 6@ L 1♀
00070: 03 00 07 00-00 00 40 12-0B 30 62 10-00 03 4B 1C ♥ • @†♂b▶ ♥KL
00080: 7F 00 04 40-C0 00 31 12-03 10 24 00-00 00 31 0C ♂ ♦@ L 1↓♥▶$ 1♀
00090: 03 00 01 C0-00 03 08 00-03 C0 0F AD-00 02 00 D2 ♥ @ L ♥ ♀ ♥ Lон ♂ П
000A0: 03 20 63 40-A9 03 FD D2-7F FC 84 00-19 00 3D C2 ♥ с@й♥дТΔN°Д ↓ =Т
000B0: 27 04 00 40-C0 00 08 D0-03 40 08 00-00 00 48 DF '♦ @ L ♀♥@ ♀ Н
000C0: 03 00 A1 C0-00 03 08 A0-7B A6 3A 83-00 01 08 F0 ♥ 6 L ♥ ♀a{ж:Г ♂♀Е
000D0: 07 0F A1 40-08 03 3F F2-03 01 01 80-1F 01 0B E0 •♂6@♀♥?€♥00A▼0♂р
000E0: 2F 95 08 C0-C0 00 3E E2-03 30 64 C0-00 00 BC EF /X ♀ L L >Т♥0d L ♀я
000F0: 03 00 41 00-00 03 40 B2-0B 2F 62 50-C0 00 3B 02 ♥ A ♥@♂/bP L ;♂
```

# Следующая цель: Intel Gemini Lake



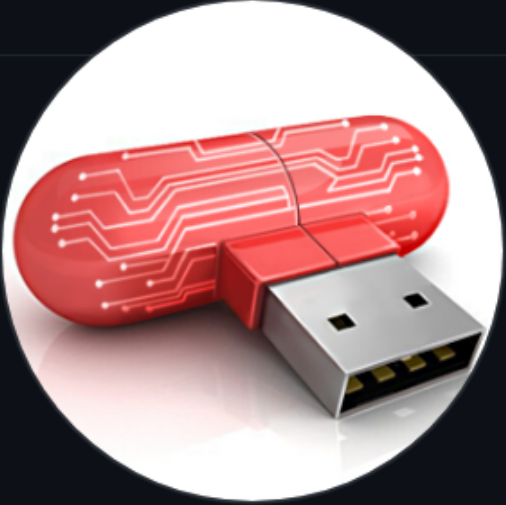
# Недокументированные инструкции

- Мы обнаружили две недокументированные инструкции, которые позволяют читать и писать во внутренние шины ЦПУ;
- Мы назвали их *udbgrd* и *udbgwr*;
- Инструкции могут быть активированы через недокументированный регистр MSR 0x1e6 (после процедуры разблокировки - Red Unlock);
- Эти инструкции позволяют напрямую изменять микрокод и обходить проверку целостности.





# Chip Red Pill



**uCode Research Team**  
chip-red-pill

Follow

Research Team Members: Dmitry Sklyarov (@\_Dmit), Mark Ermolov (@\_markel\_\_), Maxim Goryachy (@h0t)

Overview Repositories 5 Projects Packages

### Popular repositories

Repository Name	Language	Stars	Forks
<b>uCodeDisasm</b>	Python	238	24
<b>glm-ucode</b>	Python	233	37
<b>crbus_scripts</b>	Python	105	22
<b>IntelTXE-PoC</b>	Python	61	13
<b>udbgInstr</b>	C++	59	7

**uCodeDisasm**  
Python 238 stars 24 forks

**glm-ucode**  
GLM uCode dumps  
233 stars 37 forks

**crbus\_scripts**  
IPC scripts for access to Intel CRBUS  
Python 105 stars 22 forks

**IntelTXE-PoC**  
Forked from ptresearch/IntelTXE-PoC  
Intel Management Engine JTAG Proof of Concept  
Python 61 stars 13 forks

**udbgInstr**  
C++ 59 stars 7 forks

<https://github.com/chip-red-pill/>

# Intel Feedback – Bug Bounty Program

## Bug Bounty Bonus: Pentium®, Celeron®, and Intel Atom® Processors

Intel is announcing a new bonus incentive to our bug bounty program, focusing on firmware and hardware within Intel® Pentium®, Intel® Celeron®, and Intel Atom® processors (see below for full platform listing). This bonus incentive will be open to the public for a period of one year, May 11, 2021 - May 10, 2022 and will pay up to \$150,000.00 for novel vulnerabilities (1.5x the normal maximum). Additionally, at the end of the one-year period, the top 10 submissions will be identified and recognized, and the top two researchers will be invited to speak (Virtually) at iSecCon (Intel's internal security conference).

Bonus incentive open to the public – submissions must be received by 11:59pm PST on May 10, 2022 to be eligible for the bonus incentive. Submissions received after that date are not eligible for the bonus incentive but may be eligible under Intel's standard bug bounty program.

Bonus incentive award payout will be multiplier ranging from 1.2-1.5 the standing Bug Bounty payment. (See quick look chart below)

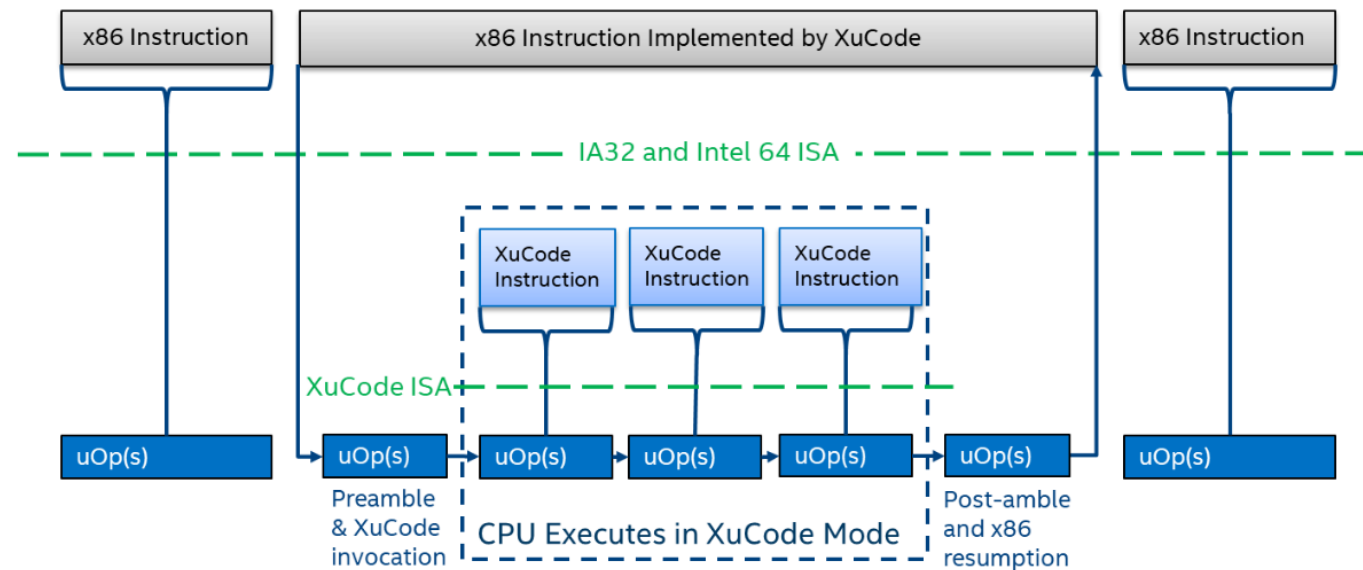
Vulnerability Severity	Intel Bug Bounty Bonus Firmware	Intel Bug Bounty Bonus Hardware
Critical	Up to \$45,000	Up to \$150,000
High	Up to \$21,000	Up to \$42,000
Medium	Up to \$3,900	Up to \$6,500
Low	Up to \$1,200	Up to \$2,400

# Intel Feedback – XuCode

## What is XuCode?

To understand what XuCode is, first we need to understand a little about microcode (sometimes written  $\mu$ code). In a Complex Instruction Set (CISC) machine like x86, the stream of instructions read from memory are decoded into small operations known as micro-ops ( $\mu$ ops). For some x86 instructions there is a simple 1:1 mapping to a  $\mu$ op, and in some cases the instruction maps to a sequence of  $\mu$ op instructions. The collection of  $\mu$ op sequences is what is often referred to as microcode. The amount of resources that can be dedicated to these microcode sequences in any given product is finite due to space and performance/design requirements.

XuCode is implemented as a variant of 64-bit mode code, running from protected system memory, using a special execution mode of the CPU. It is authenticated and loaded as part of a microcode update and is installed into a Processor Reserved Memory (PRM) range, typically allocated by system firmware. The memory range itself is protected from software and direct memory accesses by the Processor Reserved Memory Range Registers (PRMRRs). XuCode has its own set of instructions based mostly on the 64-bit Instruction Set, removing some unnecessary instructions, and adding a limited number of additional XuCode-only instructions and model specific registers (MSRs) to assist with the implementation of Intel SGX.



# Выводы

- Получили максимально возможный уровень доступа к отладочным механизмам процессоров семейства Apollo Gemini Lake;
- Восстановили большую часть микрокода;
- Обнаружили недокументированные команды, позволяющие получить доступ к внутренним шинам ЦПУ;
- Активация недокументированных команд возможна без физического доступа – через изменение прошивки BIOS;
- Разработали дизассемблер микрокода и опиасли механизм разблокировки - Chip Red Pill (<https://github.com/chip-red-pill>)

# СПАСИБО ЗА ВНИМАНИЕ!

 ***\_Dmit***      ***Dmitry Sklyarov***  
 ***h0t\_max***      ***Maxim Goryachy***  
 ***\_markel\_\_***      ***Mark Ermolov***