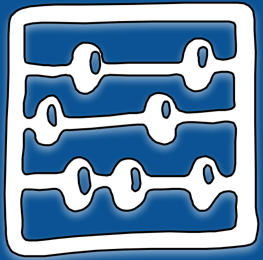
Архитектура ЭВМ и Язык Ассемблера

Семинар #4:

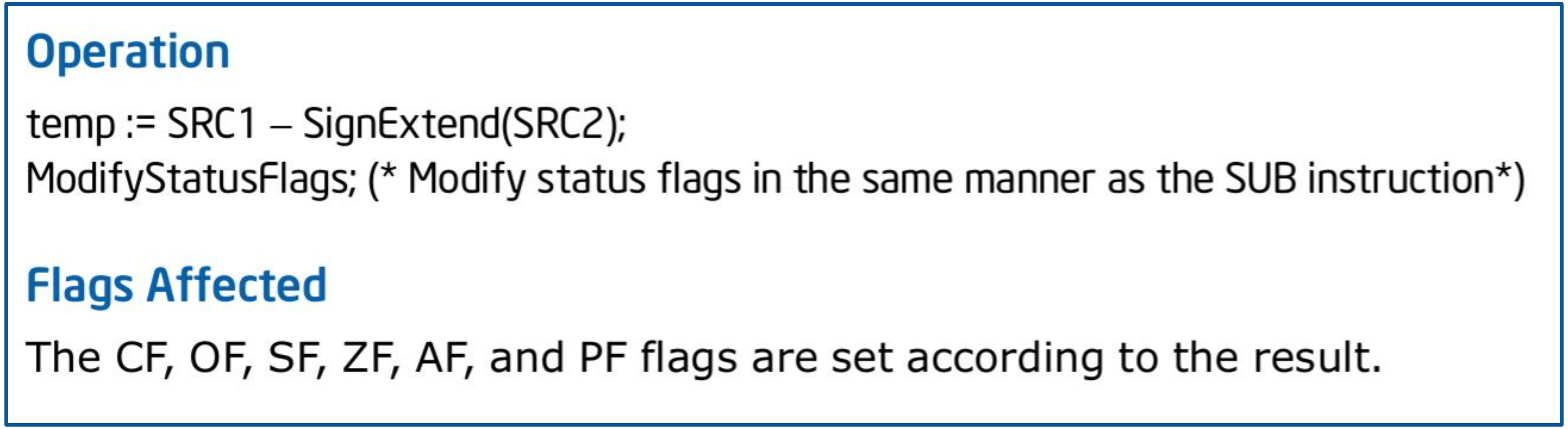
1. Применение регистра EFLAGS для сравнений.
2. Зоопарка Jcc/CMOVcc инструкций.
3. Распознавание условных ветвлений.
4. Компиляция цепочки else+if и switch+case.
5. Задача на вычисление модуля.

11.04.2024

Применение регистра EFLAGS для сравнения



Инструкция CMP и регистр EFLAGS



Operation

```
temp := SRC1 − SignExtend(SRC2);
ModifyStatusFlags; (* Modify status flags in the same manner as the SUB instruction*)
```

Flags Affected

The CF, OF, SF, ZF, AF, and PF flags are set according to the result.

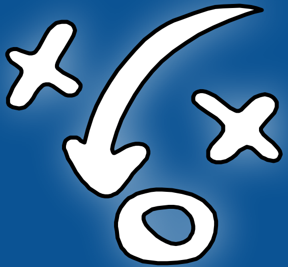
OF = “знаковое переполнение”

CF = “беззнаковое переполнение”

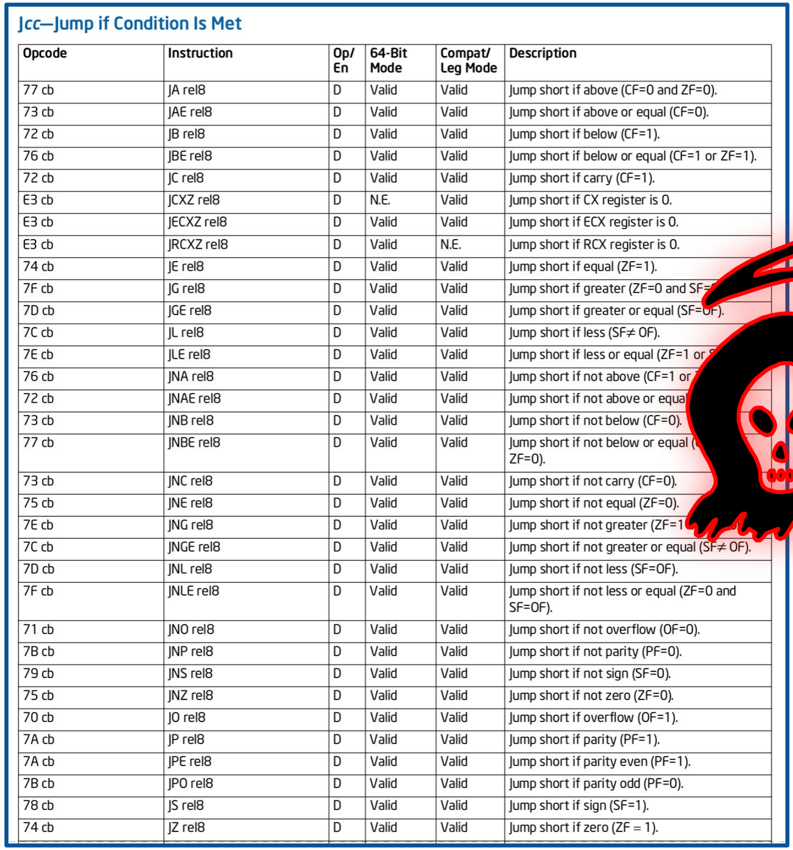
ZF = “результат равен 0”

SF = “результат отрицателен”

Зоопарк Jcc/CMOVcc инструкций

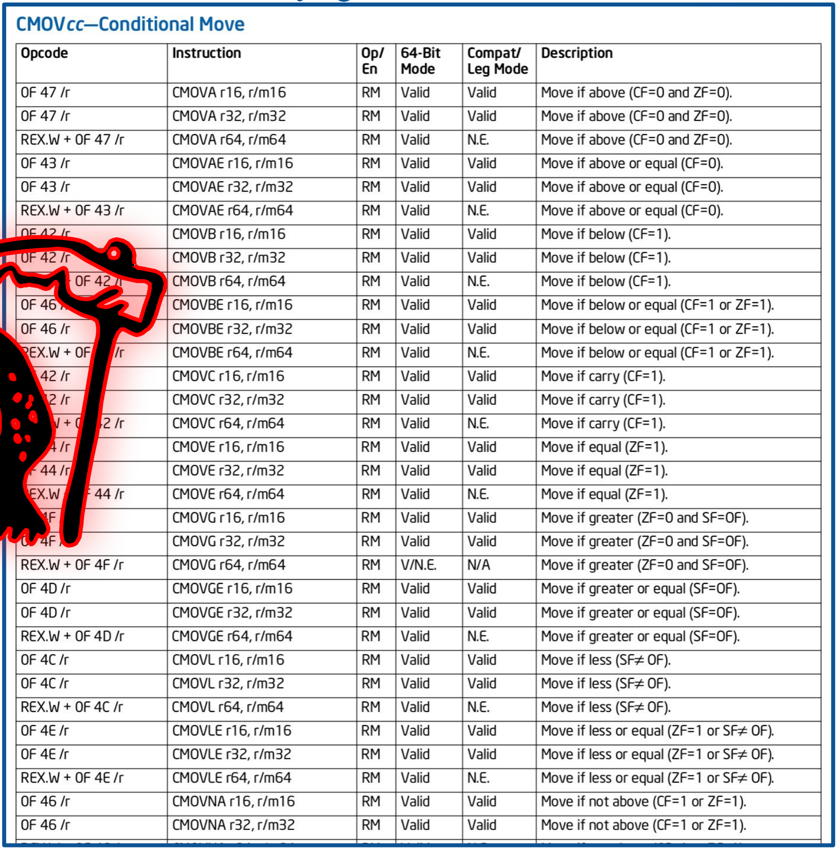


Зоопарк Jcc/CMOVcc-инструкций



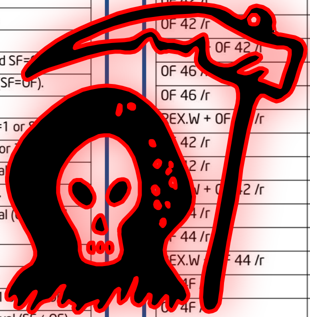
J*cc*—Jump if Condition Is Met

Opcode	Instruction	Op/En	64-Bit Mode	Compat/Leg Mode	Description
77 cb	JA rel8	D	Valid	Valid	Jump short if above (CF=0 and ZF=0).
73 cb	JAE rel8	D	Valid	Valid	Jump short if above or equal (CF=0).
72 cb	JB rel8	D	Valid	Valid	Jump short if below (CF=1).
76 cb	JBE rel8	D	Valid	Valid	Jump short if below or equal (CF=1 or ZF=1).
72 cb	JC rel8	D	Valid	Valid	Jump short if carry (CF=1).
E3 cb	JCXZ rel8	D	N.E.	Valid	Jump short if CX register is 0.
E3 cb	JECXZ rel8	D	Valid	Valid	Jump short if ECX register is 0.
E3 cb	JRCXZ rel8	D	Valid	N.E.	Jump short if RCX register is 0.
74 cb	JE rel8	D	Valid	Valid	Jump short if equal (ZF=1).
7F cb	JG rel8	D	Valid	Valid	Jump short if greater (ZF=0 and SF=
7D cb	JGE rel8	D	Valid	Valid	Jump short if greater or equal (SF=OF).
7C cb	JL rel8	D	Valid	Valid	Jump short if less (SF≠ OF).
7E cb	JLE rel8	D	Valid	Valid	Jump short if less or equal (ZF=1 or S
76 cb	JNA rel8	D	Valid	Valid	Jump short if not above (CF=1 or
72 cb	JNAE rel8	D	Valid	Valid	Jump short if not above or equal
73 cb	JNB rel8	D	Valid	Valid	Jump short if not below (CF=0).
77 cb	JNBE rel8	D	Valid	Valid	Jump short if not below or equal (ZF=0).
73 cb	JNC rel8	D	Valid	Valid	Jump short if not carry (CF=0).
75 cb	JNE rel8	D	Valid	Valid	Jump short if not equal (ZF=0).
7E cb	JNG rel8	D	Valid	Valid	Jump short if not greater (ZF=1
7C cb	JNGE rel8	D	Valid	Valid	Jump short if not greater or equal (SF≠ OF).
7D cb	JNL rel8	D	Valid	Valid	Jump short if not less (SF=OF).
7F cb	JNLE rel8	D	Valid	Valid	Jump short if not less or equal (ZF=0 and SF=OF).
71 cb	JNO rel8	D	Valid	Valid	Jump short if not overflow (OF=0).
7B cb	JNP rel8	D	Valid	Valid	Jump short if not parity (PF=0).
79 cb	JNS rel8	D	Valid	Valid	Jump short if not sign (SF=0).
75 cb	JNZ rel8	D	Valid	Valid	Jump short if not zero (ZF=0).
70 cb	JO rel8	D	Valid	Valid	Jump short if overflow (OF=1).
7A cb	JP rel8	D	Valid	Valid	Jump short if parity (PF=1).
7A cb	JPE rel8	D	Valid	Valid	Jump short if parity even (PF=1).
7B cb	JPO rel8	D	Valid	Valid	Jump short if parity odd (PF=0).
78 cb	JS rel8	D	Valid	Valid	Jump short if sign (SF=1).
74 cb	JZ rel8	D	Valid	Valid	Jump short if zero (ZF = 1).



CMOV*cc*—Conditional Move

Opcode	Instruction	Op/En	64-Bit Mode	Compat/Leg Mode	Description
0F 47 /r	CMOVA r16, r/m16	RM	Valid	Valid	Move if above (CF=0 and ZF=0).
0F 47 /r	CMOVA r32, r/m32	RM	Valid	Valid	Move if above (CF=0 and ZF=0).
REX.W + 0F 47 /r	CMOVA r64, r/m64	RM	Valid	N.E.	Move if above (CF=0 and ZF=0).
0F 43 /r	CMOVAE r16, r/m16	RM	Valid	Valid	Move if above or equal (CF=0).
0F 43 /r	CMOVAE r32, r/m32	RM	Valid	Valid	Move if above or equal (CF=0).
REX.W + 0F 43 /r	CMOVAE r64, r/m64	RM	Valid	N.E.	Move if above or equal (CF=0).
0F 42 /r	CMOVB r16, r/m16	RM	Valid	Valid	Move if below (CF=1).
0F 42 /r	CMOVB r32, r/m32	RM	Valid	Valid	Move if below (CF=1).
+ 0F 42 /r	CMOVB r64, r/m64	RM	Valid	N.E.	Move if below (CF=1).
0F 46	CMOVBE r16, r/m16	RM	Valid	Valid	Move if below or equal (CF=1 or ZF=1).
0F 46 /r	CMOVBE r32, r/m32	RM	Valid	Valid	Move if below or equal (CF=1 or ZF=1).
EX.W + 0F /r	CMOVBE r64, r/m64	RM	Valid	N.E.	Move if below or equal (CF=1 or ZF=1).
42 /r	CMOVC r16, r/m16	RM	Valid	Valid	Move if carry (CF=1).
2 /r	CMOVC r32, r/m32	RM	Valid	Valid	Move if carry (CF=1).
W + 0 2 /r	CMOVC r64, r/m64	RM	Valid	N.E.	Move if carry (CF=1).
4 /r	CMOVE r16, r/m16	RM	Valid	Valid	Move if equal (ZF=1).
44 /r	CMOVE r32, r/m32	RM	Valid	Valid	Move if equal (ZF=1).
EX.W F 44 /r	CMOVE r64, r/m64	RM	Valid	N.E.	Move if equal (ZF=1).
4F	CMOVG r16, r/m16	RM	Valid	Valid	Move if greater (ZF=0 and SF=OF).
4F	CMOVG r32, r/m32	RM	Valid	Valid	Move if greater (ZF=0 and SF=OF).
REX.W + 0F 4F /r	CMOVG r64, r/m64	RM	V/N.E.	N/A	Move if greater (ZF=0 and SF=OF).
0F 4D /r	CMOVGE r16, r/m16	RM	Valid	Valid	Move if greater or equal (SF=OF).
0F 4D /r	CMOVGE r32, r/m32	RM	Valid	Valid	Move if greater or equal (SF=OF).
REX.W + 0F 4D /r	CMOVGE r64, r/m64	RM	Valid	N.E.	Move if greater or equal (SF=OF).
0F 4C /r	CMOVL r16, r/m16	RM	Valid	Valid	Move if less (SF≠ OF).
0F 4C /r	CMOVL r32, r/m32	RM	Valid	Valid	Move if less (SF≠ OF).
REX.W + 0F 4C /r	CMOVL r64, r/m64	RM	Valid	N.E.	Move if less (SF≠ OF).
0F 4E /r	CMOVLE r16, r/m16	RM	Valid	Valid	Move if less or equal (ZF=1 or SF≠ OF).
0F 4E /r	CMOVLE r32, r/m32	RM	Valid	Valid	Move if less or equal (ZF=1 or SF≠ OF).
REX.W + 0F 4E /r	CMOVLE r64, r/m64	RM	Valid	N.E.	Move if less or equal (ZF=1 or SF≠ OF).
0F 46 /r	CMOVNA r16, r/m16	RM	Valid	Valid	Move if not above (CF=1 or ZF=1).
0F 46 /r	CMOVNA r32, r/m32	RM	Valid	Valid	Move if not above (CF=1 or ZF=1).

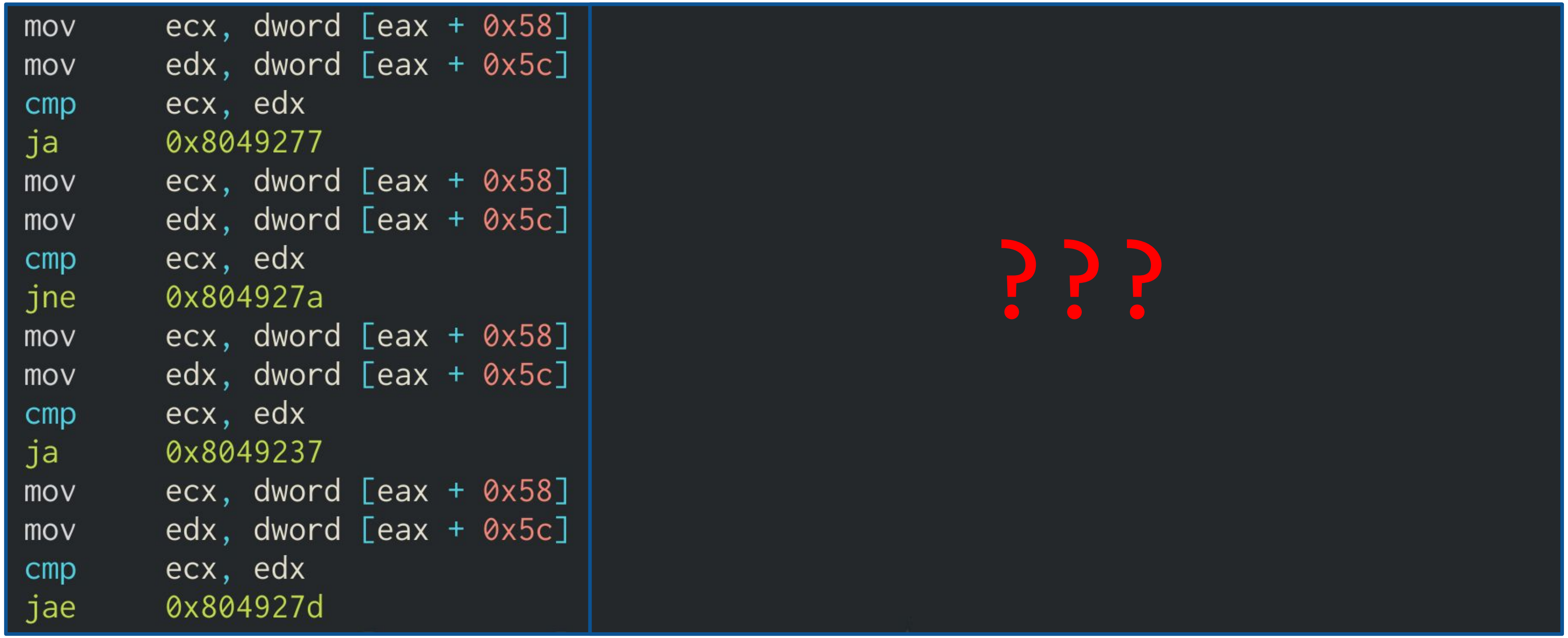


Зоопарк Jcc/CMOVcc-инструкций

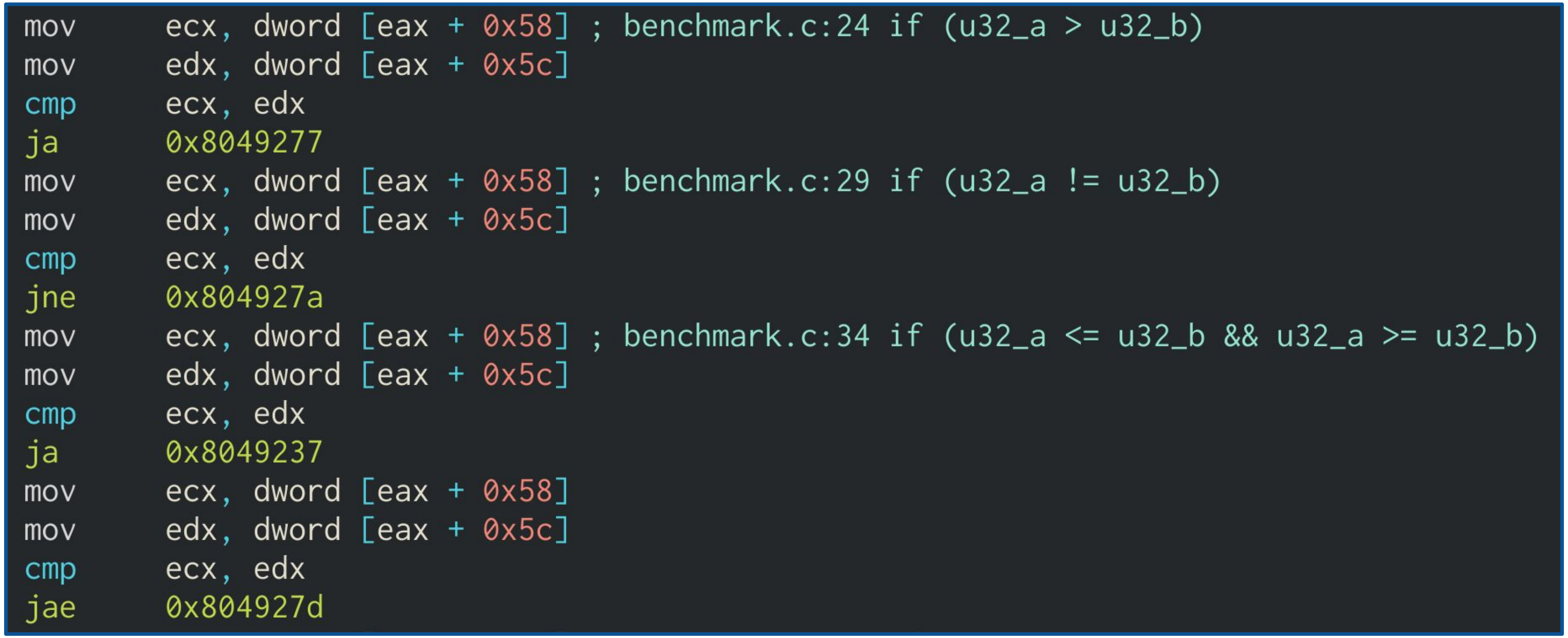
КОП		Флаги	Описание
JO		$OF = 1$	overflow
JS		$SF = 1$	sign
JZ	JE	$ZF = 1$	zero/equal
JC	JB	$CF = 1$	below
	JBE	$CF = 1$ или $ZF = 1$	below or equal
JNC	JAE	$CF = 0$	above or equal
	JA	$CF = 0$ и $ZF = 0$	above
	JL	$SF \neq OF$	less
	JLE	$SF \neq OF$ или $ZF = 1$	less or equal
	JGE	$SF = OF$	greater or equal
	JG	$SF = OF$ и $ZF = 0$	greater
JECXZ		$ECX = 0$	

Распознавание условных ветвлений

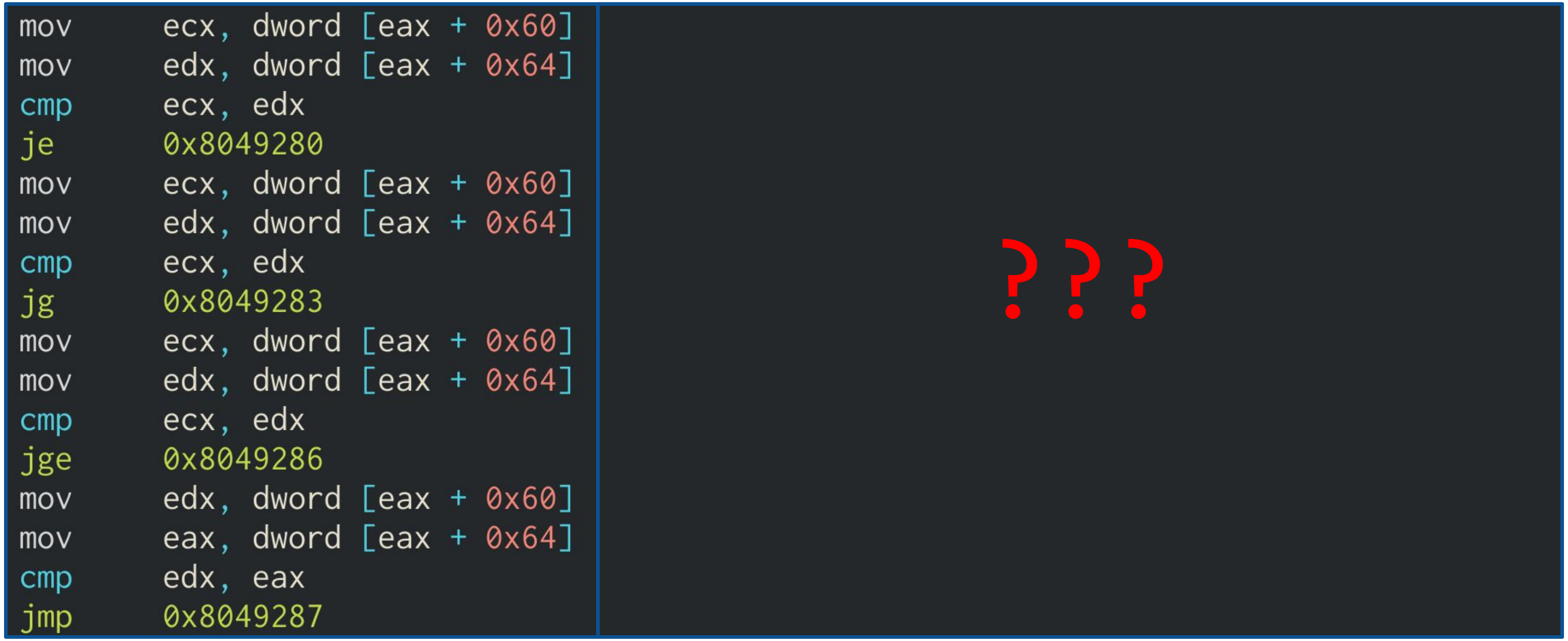
Распознавание условных ветвлений


```
mov     ecx, dword [eax + 0x58]
mov     edx, dword [eax + 0x5c]
cmp     ecx, edx
ja      0x8049277
mov     ecx, dword [eax + 0x58]
mov     edx, dword [eax + 0x5c]
cmp     ecx, edx
jne     0x804927a
mov     ecx, dword [eax + 0x58]
mov     edx, dword [eax + 0x5c]
cmp     ecx, edx
ja      0x8049237
mov     ecx, dword [eax + 0x58]
mov     edx, dword [eax + 0x5c]
cmp     ecx, edx
jae     0x804927d
```

Распознавание условных ветвлений

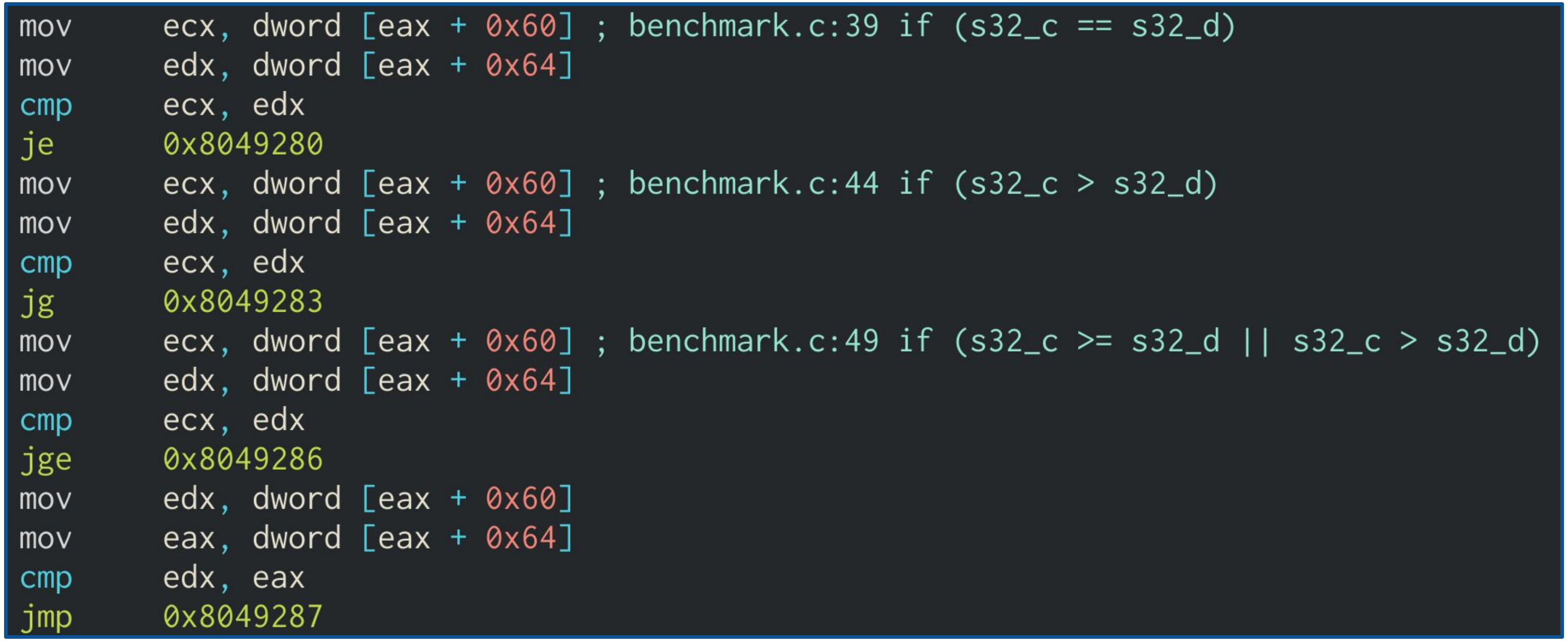

```
mov      ecx, dword [eax + 0x58] ; benchmark.c:24 if (u32_a > u32_b)
mov      edx, dword [eax + 0x5c]
cmp      ecx, edx
ja       0x8049277
mov      ecx, dword [eax + 0x58] ; benchmark.c:29 if (u32_a != u32_b)
mov      edx, dword [eax + 0x5c]
cmp      ecx, edx
jne      0x804927a
mov      ecx, dword [eax + 0x58] ; benchmark.c:34 if (u32_a <= u32_b && u32_a >= u32_b)
mov      edx, dword [eax + 0x5c]
cmp      ecx, edx
ja       0x8049237
mov      ecx, dword [eax + 0x58]
mov      edx, dword [eax + 0x5c]
cmp      ecx, edx
jae      0x804927d
```

Распознавание условных ветвлений



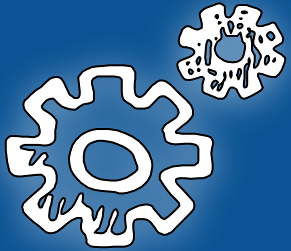
```
mov     ecx, dword [eax + 0x60]
mov     edx, dword [eax + 0x64]
cmp     ecx, edx
je      0x8049280
mov     ecx, dword [eax + 0x60]
mov     edx, dword [eax + 0x64]
cmp     ecx, edx
jg      0x8049283
mov     ecx, dword [eax + 0x60]
mov     edx, dword [eax + 0x64]
cmp     ecx, edx
jge     0x8049286
mov     edx, dword [eax + 0x60]
mov     eax, dword [eax + 0x64]
cmp     edx, eax
jmp     0x8049287
```

Распознавание условных ветвлений

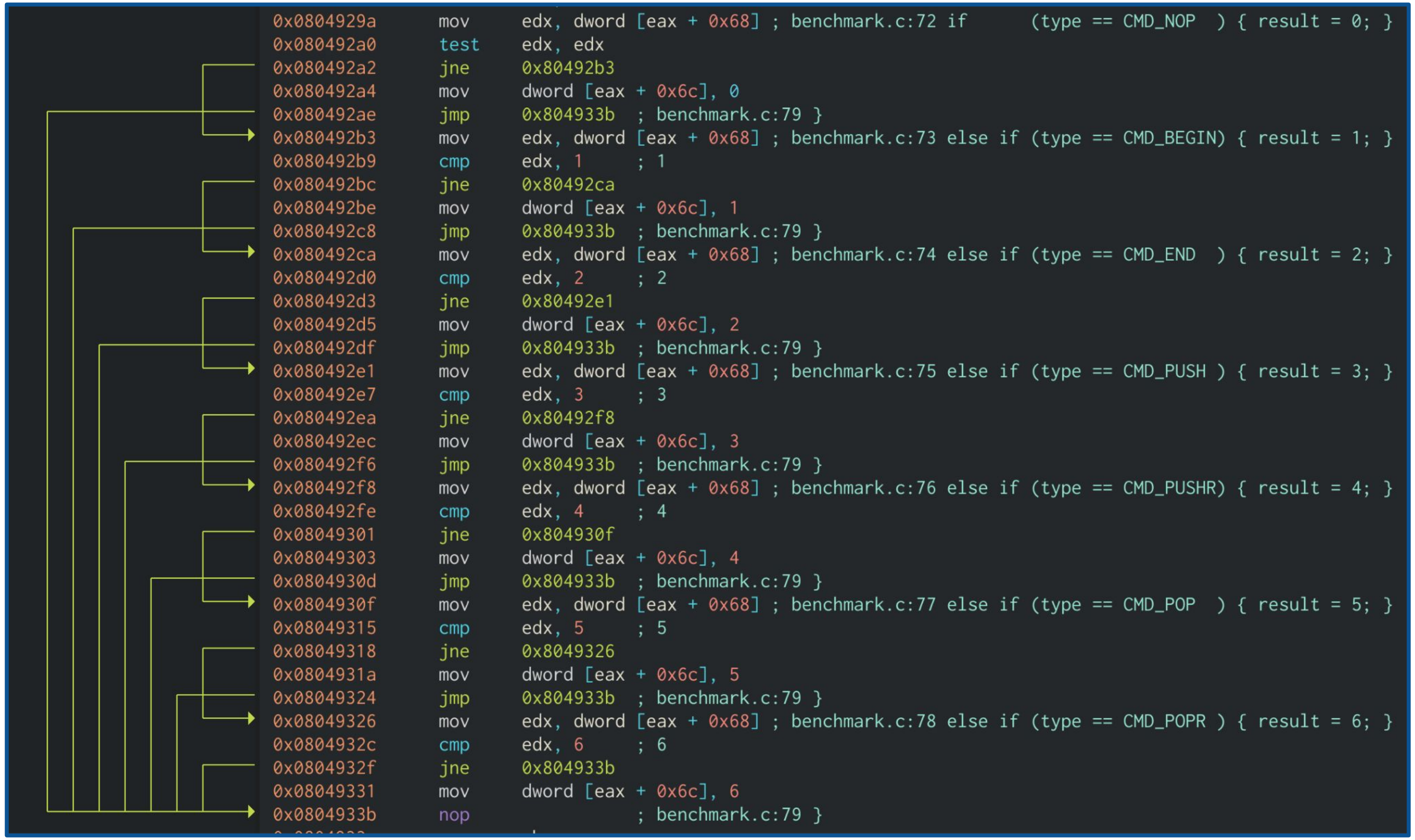


```
mov     ecx, dword [eax + 0x60] ; benchmark.c:39 if (s32_c == s32_d)
mov     edx, dword [eax + 0x64]
cmp     ecx, edx
je      0x8049280
mov     ecx, dword [eax + 0x60] ; benchmark.c:44 if (s32_c > s32_d)
mov     edx, dword [eax + 0x64]
cmp     ecx, edx
jg      0x8049283
mov     ecx, dword [eax + 0x60] ; benchmark.c:49 if (s32_c >= s32_d || s32_c > s32_d)
mov     edx, dword [eax + 0x64]
cmp     ecx, edx
jge     0x8049286
mov     edx, dword [eax + 0x60]
mov     eax, dword [eax + 0x64]
cmp     edx, eax
jmp     0x8049287
```

Компиляция цепочки else+if и switch+case

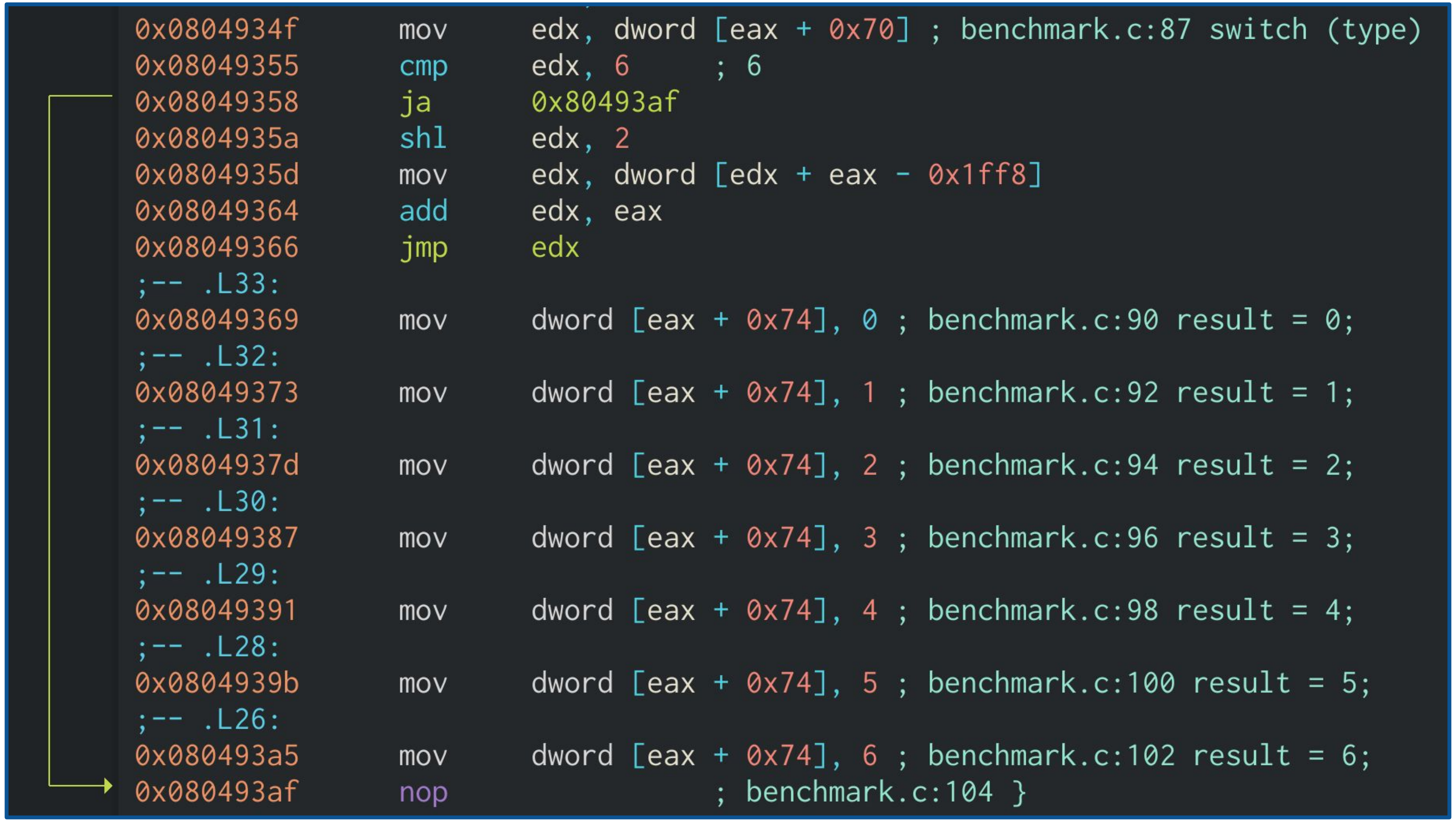


Компиляция цепочки else+if



```
0x0804929a    mov    edx, dword [eax + 0x68] ; benchmark.c:72 if      (type == CMD_NOP  ) { result = 0; }
0x080492a0    test   edx, edx
0x080492a2    jne    0x80492b3
0x080492a4    mov    dword [eax + 0x6c], 0
0x080492ae    jmp    0x804933b  ; benchmark.c:79 }
0x080492b3    mov    edx, dword [eax + 0x68] ; benchmark.c:73 else if (type == CMD_BEGIN) { result = 1; }
0x080492b9    cmp    edx, 1     ; 1
0x080492bc    jne    0x80492ca
0x080492be    mov    dword [eax + 0x6c], 1
0x080492c8    jmp    0x804933b  ; benchmark.c:79 }
0x080492ca    mov    edx, dword [eax + 0x68] ; benchmark.c:74 else if (type == CMD_END  ) { result = 2; }
0x080492d0    cmp    edx, 2     ; 2
0x080492d3    jne    0x80492e1
0x080492d5    mov    dword [eax + 0x6c], 2
0x080492df    jmp    0x804933b  ; benchmark.c:79 }
0x080492e1    mov    edx, dword [eax + 0x68] ; benchmark.c:75 else if (type == CMD_PUSH ) { result = 3; }
0x080492e7    cmp    edx, 3     ; 3
0x080492ea    jne    0x80492f8
0x080492ec    mov    dword [eax + 0x6c], 3
0x080492f6    jmp    0x804933b  ; benchmark.c:79 }
0x080492f8    mov    edx, dword [eax + 0x68] ; benchmark.c:76 else if (type == CMD_PUSHR) { result = 4; }
0x080492fe    cmp    edx, 4     ; 4
0x08049301    jne    0x804930f
0x08049303    mov    dword [eax + 0x6c], 4
0x0804930d    jmp    0x804933b  ; benchmark.c:79 }
0x0804930f    mov    edx, dword [eax + 0x68] ; benchmark.c:77 else if (type == CMD_POP  ) { result = 5; }
0x08049315    cmp    edx, 5     ; 5
0x08049318    jne    0x8049326
0x0804931a    mov    dword [eax + 0x6c], 5
0x08049324    jmp    0x804933b  ; benchmark.c:79 }
0x08049326    mov    edx, dword [eax + 0x68] ; benchmark.c:78 else if (type == CMD_POPR ) { result = 6; }
0x0804932c    cmp    edx, 6     ; 6
0x0804932f    jne    0x804933b
0x08049331    mov    dword [eax + 0x6c], 6
0x0804933b    nop               ; benchmark.c:79 }
```

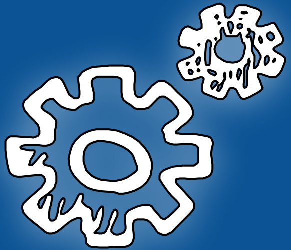
Компиляция цепочки switch+case



```
0x0804934f      mov     edx, dword [eax + 0x70] ; benchmark.c:87 switch (type)
0x08049355      cmp     edx, 6      ; 6
0x08049358      ja      0x80493af
0x0804935a      shl     edx, 2
0x0804935d      mov     edx, dword [edx + eax - 0x1ff8]
0x08049364      add     edx, eax
0x08049366      jmp     edx
;-- .L33:
0x08049369      mov     dword [eax + 0x74], 0 ; benchmark.c:90 result = 0;
;-- .L32:
0x08049373      mov     dword [eax + 0x74], 1 ; benchmark.c:92 result = 1;
;-- .L31:
0x0804937d      mov     dword [eax + 0x74], 2 ; benchmark.c:94 result = 2;
;-- .L30:
0x08049387      mov     dword [eax + 0x74], 3 ; benchmark.c:96 result = 3;
;-- .L29:
0x08049391      mov     dword [eax + 0x74], 4 ; benchmark.c:98 result = 4;
;-- .L28:
0x0804939b      mov     dword [eax + 0x74], 5 ; benchmark.c:100 result = 5;
;-- .L26:
0x080493a5      mov     dword [eax + 0x74], 6 ; benchmark.c:102 result = 6;
0x080493af      nop                     ; benchmark.c:104 }
```

HSE University

Задача на вычисление модуля



Вопросы?

Красивые иконки взяты с сайта handdrawngoods.com