

Архитектура ЭВМ и язык ассемблера

Семинар #31:

1. Условные ветвления.
2. Простейшая программа с ветвлениями.
3. Изучение кодогенерации: `if+else` vs `switch`

Условные ветвления



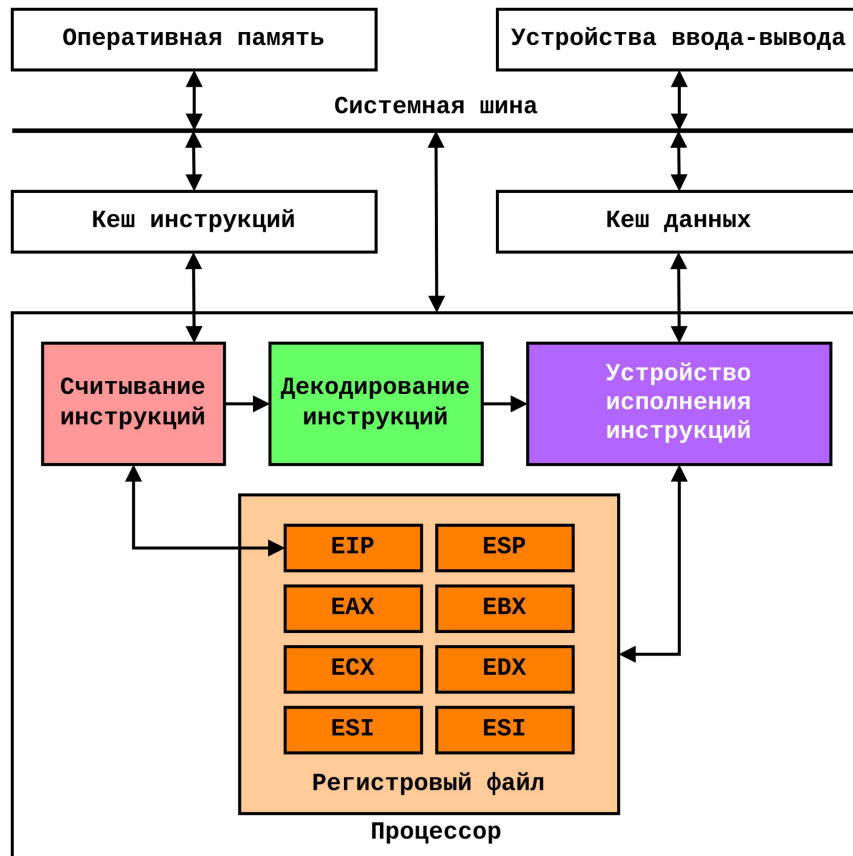
Инструкции для условных ветвлений

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
JNC JAE	CF = 0	above or equal
	JA	CF = 0 и ZF = 0
	JL	SF \neq OF
	JLE	SF \neq OF или ZF = 1
	JGE	SF = OF
	JG	SF = OF и ZF = 0
JECXZ	ECX = 0	

Простейшая модель процессора

Алгоритм работы процессора:

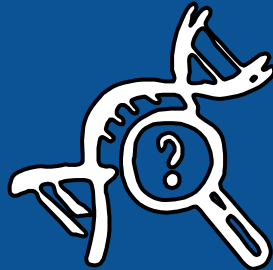
1. Fetch Instruction.
2. Decode instruction.
3. Fetch operands.
4. Execute instruction:
 - Обновление регистров.
 - Чтение из памяти.
5. Write result to cache.



Простейшая программа с ветвлениями



Изучение кодогенерации: if+else vs switch



Кодогенерация условных переходов

```
void example_conditionals(void)
{
    static uint32_t u32_a, u32_b;
    static int32_t s32_c, s32_d;

    if (u32_a > u32_b)
    {
        return;
    }

    if (u32_a != u32_b)
    {
        return;
    }

    if (u32_a <= u32_b && u32_a >= u32_b)
    {
        return;
    }

    if (s32_c == s32_d)
    {
        return;
    }

    if (s32_c > s32_d)
    {
        return;
    }

    if (s32_c >= s32_d || s32_c < s32_d)
    {
        return;
    }
}
```

```

80491f7:      8b 88 44 00 00 00      mov     0x44(%eax),%ecx
80491fd:      8b 90 48 00 00 00      mov     0x48(%eax),%edx
8049203:      39 d1                    cmp     %edx,%ecx
8049205:      77 70                    ja      8049277 <example_conditionals+0x91>
8049207:      8b 88 44 00 00 00      mov     0x44(%eax),%ecx
804920d:      8b 90 48 00 00 00      mov     0x48(%eax),%edx
8049213:      39 d1                    cmp     %edx,%ecx
8049215:      75 63                    jne     804927a <example_conditionals+0x94>
8049217:      8b 88 44 00 00 00      mov     0x44(%eax),%ecx
804921d:      8b 90 48 00 00 00      mov     0x48(%eax),%edx
8049223:      39 d1                    cmp     %edx,%ecx
8049225:      77 10                    ja      8049237 <example_conditionals+0x51>
8049227:      8b 88 44 00 00 00      mov     0x44(%eax),%ecx
804922d:      8b 90 48 00 00 00      mov     0x48(%eax),%edx
8049233:      39 d1                    cmp     %edx,%ecx
8049235:      73 46                    jae     804927d <example_conditionals+0x97>
8049237:      8b 88 4c 00 00 00      mov     0x4c(%eax),%ecx
804923d:      8b 90 50 00 00 00      mov     0x50(%eax),%edx
8049243:      39 d1                    cmp     %edx,%ecx
8049245:      74 39                    je      8049280 <example_conditionals+0x9a>
8049247:      8b 88 4c 00 00 00      mov     0x4c(%eax),%ecx
804924d:      8b 90 50 00 00 00      mov     0x50(%eax),%edx
8049253:      39 d1                    cmp     %edx,%ecx
8049255:      7f 2c                    jg      8049283 <example_conditionals+0x9d>
8049257:      8b 88 4c 00 00 00      mov     0x4c(%eax),%ecx
804925d:      8b 90 50 00 00 00      mov     0x50(%eax),%edx
8049263:      39 d1                    cmp     %edx,%ecx
8049265:      7d 1f                    jge     8049286 <example_conditionals+0xa0>
8049267:      8b 90 4c 00 00 00      mov     0x4c(%eax),%edx
804926d:      8b 80 50 00 00 00      mov     0x50(%eax),%eax
8049273:      39 c2                    cmp     %eax,%edx
8049275:      eb 10                    jmp     8049287 <example_conditionals+0xa1>
8049277:      nop
8049278:      eb 0d                    jmp     8049287 <example_conditionals+0xa1>
804927a:      eb 09                    jmp     8049287 <example_conditionals+0xa1>
804927b:      eb 0a                    jmp     8049287 <example_conditionals+0xa1>
804927d:      eb 09                    jmp     8049287 <example_conditionals+0xa1>
804927e:      eb 07                    jmp     8049287 <example_conditionals+0xa1>
8049280:      eb 04                    jmp     8049287 <example_conditionals+0xa1>
8049283:      eb 04                    jmp     8049287 <example_conditionals+0xa1>
8049284:      eb 01                    jmp     8049287 <example_conditionals+0xa1>
8049286:      eb 01                    jmp     8049287 <example_conditionals+0xa1>
8049287:      5d                        pop     %ebp
8049288:      c3                        ret

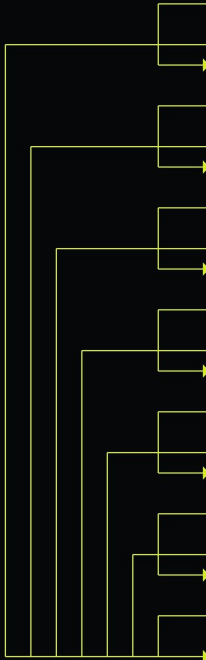
```

Паттерн: цепочка if+else

```
void example_ifelse_chain(void)
{
    static OperationType op;
    static int result;

    if      (op == OP_ADD) { result = 0; }
    else if (op == OP_SUB) { result = 1; }
    else if (op == OP_MUL) { result = 2; }
    else if (op == OP_DIV) { result = 3; }
    else if (op == OP_SIN) { result = 4; }
    else if (op == OP_COS) { result = 5; }
    else if (op == OP_EXP) { result = 6; }
}
```


Паттерн: цепочка if+else



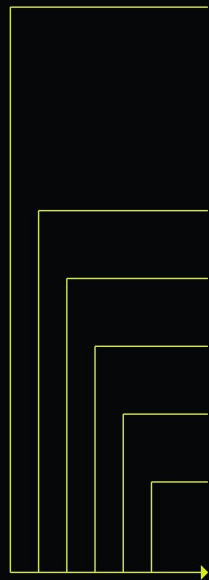
The diagram illustrates the 'if-else chain' pattern in assembly code. It shows a series of memory addresses on the left, with yellow arrows indicating the flow of execution. The flow starts at 0x08049289, goes down to 0x080492a0, then branches to 0x080492a2, 0x080492ae, 0x080492b3, 0x080492bc, 0x080492be, 0x080492c8, 0x080492ca, 0x080492d3, 0x080492d5, 0x080492df, 0x080492e1, 0x080492e7, 0x080492ea, 0x080492ec, 0x080492f6, 0x080492f8, 0x080492fe, 0x08049301, 0x08049303, 0x0804930d, 0x0804930f, 0x08049315, 0x08049318, 0x0804931a, 0x08049324, 0x08049326, 0x0804932c, 0x0804932f, 0x08049331, 0x0804933b, and finally 0x0804933c. The corresponding assembly code on the right shows a series of 'jne' (jump if not equal) instructions that branch to different labels (0x80492b3, 0x80492ae, 0x80492c8, 0x80492d3, 0x80492e1, 0x80492f6, 0x80492fe, 0x8049301, 0x804930d, 0x8049318, 0x8049324, 0x804932f, 0x8049331) based on the result of various comparisons (edx, dx, eax, etc.). The code is annotated with comments indicating the corresponding C code (benchmark.c:63, benchmark.c:67, benchmark.c:74, benchmark.c:68, benchmark.c:69, benchmark.c:70, benchmark.c:71, benchmark.c:72, benchmark.c:73, benchmark.c:74).

```
;- example_ifelse_chain:
dbg.example_ifelse_chain ();
0x08049289    endbr32                ; benchmark.c:63 { ; void example_ifelse_chain();
0x0804928d    push ebp
0x0804928e    mov ebp, esp
0x08049290    call __x86.get_pc_thunk.ax ; sym.__x86.get_pc_thunk.ax
0x08049295    add eax, 0x2d6b
0x0804929a    mov edx, dword [eax + 0x54] ; benchmark.c:67 if      (op == OP_ADD) {
0x080492a0    test edx, edx
0x080492a2    jne 0x80492b3
0x080492a4    mov dword [eax + 0x58], 0
0x080492ae    jmp 0x804933b                ; benchmark.c:74 }
0x080492b3    mov edx, dword [eax + 0x54] ; benchmark.c:68 else if (op == OP_SUB) {
0x080492b9    cmp edx, 1                    ; 1
0x080492bc    jne 0x80492ca
0x080492be    mov dword [eax + 0x58], 1
0x080492c8    jmp 0x804933b                ; benchmark.c:74 }
0x080492ca    mov edx, dword [eax + 0x54] ; benchmark.c:69 else if (op == OP_MUL) {
0x080492d0    cmp edx, 2                    ; 2
0x080492d3    jne 0x80492e1
0x080492d5    mov dword [eax + 0x58], 2
0x080492df    jmp 0x804933b                ; benchmark.c:74 }
0x080492e1    mov edx, dword [eax + 0x54] ; benchmark.c:70 else if (op == OP_DIV) {
0x080492e7    cmp edx, 3                    ; 3
0x080492ea    jne 0x80492f8
0x080492ec    mov dword [eax + 0x58], 3
0x080492f6    jmp 0x804933b                ; benchmark.c:74 }
0x080492f8    mov edx, dword [eax + 0x54] ; benchmark.c:71 else if (op == OP_SIN) {
0x080492fe    cmp edx, 4                    ; 4
0x08049301    jne 0x804930f
0x08049303    mov dword [eax + 0x58], 4
0x0804930d    jmp 0x804933b                ; benchmark.c:74 }
0x0804930f    mov edx, dword [eax + 0x54] ; benchmark.c:72 else if (op == OP_COS) {
0x08049315    cmp edx, 5                    ; 5
0x08049318    jne 0x8049326
0x0804931a    mov dword [eax + 0x58], 5
0x08049324    jmp 0x804933b                ; benchmark.c:74 }
0x08049326    mov edx, dword [eax + 0x54] ; benchmark.c:73 else if (op == OP_EXP) {
0x0804932c    cmp edx, 6                    ; 6
0x0804932f    jne 0x804933b
0x08049331    mov dword [eax + 0x58], 6
0x0804933b    nop                          ; benchmark.c:74 }
0x0804933c    pop ebp
```

Паттерн: switch

```
void example_switch(void)
{
    static OperationType type;
    static int result;

    switch (type)
    {
        case OP_ADD:
            result = 0;
        case OP_SUB:
            result = 1;
            break;
        case OP_MUL:
            result = 2;
            break;
        case OP_DIV:
            result = 3;
            break;
        case OP_SIN:
            result = 4;
            break;
        case OP_COS:
            result = 5;
            break;
        case OP_EXP:
            result = 6;
            break;
    }
}
```



```
-- example_switch:
dbg.example_switch ();
0x0804933e    endbr32                ; benchmark.c:78 { ; void example_switch();
0x08049342    push ebp
0x08049343    mov ebp, esp
0x08049345    call __x86.get_pc_thunk.ax ; sym.__x86.get_pc_thunk.ax
0x0804934a    add eax, 0x2cb6
0x0804934f    mov edx, dword [eax + 0x5c] ; benchmark.c:82 switch (type)
0x08049355    cmp edx, 6              ; 6
0x08049358    ja 0x80493ba
0x0804935a    shl edx, 2
0x0804935d    mov edx, dword [edx + eax - 0x1ff8]
0x08049364    add edx, eax
0x08049366    jmp edx

;-- .L33:
0x08049369    mov dword [eax + 0x60], 0 ; benchmark.c:85 result = 0;
;-- .L32:
0x08049373    mov dword [eax + 0x60], 1 ; benchmark.c:87 result = 1;
0x0804937d    jmp 0x80493ba            ; benchmark.c:88 break; ; dbg.example_switch+0x7c

;-- .L31:
0x0804937f    mov dword [eax + 0x60], 2 ; benchmark.c:90 result = 2;
0x08049389    jmp 0x80493ba            ; benchmark.c:91 break; ; dbg.example_switch+0x7c

;-- .L30:
0x0804938b    mov dword [eax + 0x60], 3 ; benchmark.c:93 result = 3;
0x08049395    jmp 0x80493ba            ; benchmark.c:94 break; ; dbg.example_switch+0x7c

;-- .L29:
0x08049397    mov dword [eax + 0x60], 4 ; benchmark.c:96 result = 4;
0x080493a1    jmp 0x80493ba            ; benchmark.c:97 break; ; dbg.example_switch+0x7c

;-- .L28:
0x080493a3    mov dword [eax + 0x60], 5 ; benchmark.c:99 result = 5;
0x080493ad    jmp 0x80493ba            ; benchmark.c:100 break; ; dbg.example_switch+0x7c

;-- .L26:
0x080493af    mov dword [eax + 0x60], 6 ; benchmark.c:102 result = 6;
0x080493b9    nop                    ; benchmark.c:103 break;
0x080493ba    nop                    ; benchmark.c:105 }
0x080493bb    pop ebp
0x080493bc    ret
```

Вопросы?



Красивые иконки взяты с сайта handdrawngoods.com