









ЛЕКЦИЯ 6

ВИДОВЕ ОПЕРАЦИИ

-  **Формат на инструкциите**
-  **Регистър на условията**
-  **Пренос на данни**
-  **Аритметични операции**
-  **Операции с един операнд**
-  **Логически операции**
-  **Измествания и ротации**
-  **Управление на програмата**

ФОРМАТ НА МИ

Операциите, които реализират ЦП, имат **много общи черти**.

Това е **результат от исторически и преди всичко от практически причини**.

Операциите се **класифицират по броя** на техните **операнди**: с един и с два операнда.

МИ с **1 операнд** имат следния формат:

ОП приемник и реализират **$nr := f(nr)$** .

За целта става **четене на nr** , използване на **АЛУ** за извършване на **пресмятането** и **запис** на резултата **в приемника**.

МИ С ДВА ОПЕРАНДА

МИ с **2 операнда** имат следния формат:

ОП приемник, източник.

Тези МИ реализират $pr := f(pr, изт)$.

За целта става **четене на pr , четене на $изт$** , използване на **АЛУ за извършване на пресмятането и запис на резултата в pr** .

В някои ЦП **редът е обратен: ОП $изт, pr$** .

При **акумулаторен ЦП** обикновено имаме:

ОП $изт$ $acc := f(acc, изт)$ 1-адресни.

При **ЦП с РОП** обикновено имаме:

ОП $рег, изт$ $рег := f(рег, изт)$ 1½-адресни.

ДВА ОПЕРАНДА (прод.)

При повечето РОП има и **инверсна форма**:

ОП nr, reg $nr := f(nr, reg)$ 1^{1/2}-адресни.

При **2-адресни МИ** може да бъде разрешено и двата операнда да бъдат в ОП и такива МИ реализират действия от типа **памет – памет**.

В повечето случаи (I80x86) **единият** операнд задължително се задава **само с регистрова адресация** и няма действия **памет – памет**.

Дължината на двата операнда трябва да бъде **една и съща**. Тя се определя от **КОП** на МИ.

РЕГИСТЪР НА УСЛОВИЯТА

В реалните машинни програми **МИ** за разклонение (**условен преход**) обикновено са **значителната част (до 20%)** от всички МИ.

При ЕП условният оператор има вида **If <условие> Then** като условието най-често изисква **сравняване на две величини**.

Аналогичната **МИ** трябва да бъде **3-адресна**, което **не е рационално**, защото **$a ? b$**

е еквивалентно на **$a - b ? 0$** , а често са необходими **и други** оценки на резултата.

За пренос на подобни сведения между 2 МИ се използва **регистър**, наречен **на условията**.

ИЗРАБОТКА НА РУ

Регистърът на Условиата (РУ) – Condition Code Register (CCR), може да бъде изработен по **два начина**:

- 1 n бита с 2^n стойности като всяка от тях посочва едно от **взаимно изключващите** се условия;
- 2 съвкупност от n бита като всеки регистрира **наличие или отсъствие** на **определено условие**, което е **независимо от останалите** условия.

Примери:

- 1 **IBM 360** 2-битов с **4 стойности**: **0** означава резултат **=0**, **1** – **<0**, **2** – **>0**, а **3** е **препълване**.
- 2 при **съвременните микропроцесори**.

ОСНОВНИ УСЛОВИЯ

Броят, названието и предназначението на разрядите за условия са различни, но най-разпространени са следните флагове за условия (PDP-11, M68000, I80x86 и др.):

- ① нулев резултат (**Zero, ZFlag**): $1 \leftrightarrow =0, 0 \leftrightarrow \neq 0$;
- ② знак, отрицателен резултат (**Sign, Negative, SF**);
- ③ препълване (**oVerflow, OF**) при числа със знак;
- ④ пренос (**Carry, CF**) = препълване без знак.

Установяването на флаговете се съобразява с размера на данните: при байтове N е 7-ият бит на резултата, при двойки байтове – 15-ият бит, а при четворки байтове – 31-ят бит.

ДОПЪЛНИТЕЛНИ УСЛОВИЯ

Регистрите на ЦП, вкл. **РУ**, рядко са **4-битови**.

Допълнителните битове на РУ регистрират **нови условия или управляват ЦП**. Такива са:

- ⑤ **четност (Parity, PF)** на единиците в резултата;
- ⑥ **разширение (eXtension)** като **C**, но не винаги;
- ⑦ **полупренос (Half carry)** между **3-ти и 4-ти** бит;
- ⑧ **десетична аритметика (Decimal)** при **6502**;
- ⑨ **посока (Direction)** при **I80x86** напред или назад;
- ⑩ **последна операция (Decimal correction)** + или –.

РАБОТА С РУ

Често възниква необходимост за **записване** на определена **стойност** в някой от **флаговете** на РУ. За целта има **специални МИ**:

CLRC C:=0; CLR V:=0; CLRN N:=0; CLRZ Z:=0;
SETC C:=1; SET V:=1; SETN N:=1; SETZ Z:=1;

Понякога има **МИ за работа с РУ** като цяло:

ANDCC – **логическо И** с РУ (нулиране);

ORCC – **логическо ИЛИ** с РУ (запис единици);

PUSHCC – **запис** на РУ **в стека**;

PULLCC – **четене** на РУ **от стека**.

МАШИНЕН ЕЗИК

От съществено значение при създаване на МЕ е **правилното кодиране** на операциите.

За тази цел се използват сведения за това **кои операции се използват по-често при писането** на програми.

Такива операции се реализират **с по-богат набор от адресации** на операндите.

За **оптимизиране** на изработката на ЦП са полезни и сведения за това **кои операции се изпълняват по-често** от него.

ПРЕНОС НА ДАННИ

Това е **най-често използваната група** при създаване на машинни програми.

Тази група предизвиква и **най-големите спорове** по отношение на **РУ**: за **Z** и **N** почти няма спор, но за **C** и **V** се прилагат различни схеми – **нулиране** или **без промяна**.

Обичайните **названия** са:

LD (LoaD) зареждане на **регистър от ОП**;

ST (STore) запис на **регистър в ОП**;

MOV (MOVe) **пренос**: при **I80x86** единият операнд **е регистър**, при **PDP-11** и **M6800** са допустими и **преноси памет-памет**.

СПЕЦИАЛНИ ПРЕНОСИ

За работа със **стек**:

PDP-11, M6809, M68000 – **ST -(R)** и **LD (R)+**;

Z8000 – специални МИ с **всеки регистър**;

I80x86 – специални МИ, но **само спрямо SP**;

M6809 – **PSHS** и **PULS** няколко регистъра;

M68000 – **LDM** и **STM** няколко регистъра.

Регистър – регистър:

TFR (TransFeR) и **XCH (eXCHange)** – размяна;

SWAP – размяна на части от регистър.

Масов обмен на регистри с ОП: LDM и STM.

Блок памет: I80x86 (през A) и др.

АРИТМЕТИЧНИ ОПЕРАЦИИ

Стандартна аритметика:

Събиране – **ADD** пр,изт $пр := \sum(пр, изт, 0)$;
 Изваждане – **SUB** пр,изт $пр := \sum(пр, \neg изт, 1)$
 (**SUBtract**);
 Сравняване – **CMR** пр,изт $\sum(пр, \neg изт, 1)$
 (**CoMPare**).

Аритметика с повишена точност:

Събиране с пренос – **ADC** $пр := пр + изт + C$;
 Изваждане със заем – **SBC** $пр := пр - изт - C$
 (**ADd/SuBtract with Carry**).
 Размножаване на знак – **SXT** (**Sign eXTend**).

УМНОЖЕНИЕ И ДЕЛЕНЕ

Повечето процесори днес имат и инструкции за **умножение и деление** на числа **със знак, без знак** или **и за двата вида** числа.

Особеното при тези операции е, че **произведението**, а от там **и делимото**, ще бъде **с удвоена точност** (брой бита).

MUL R_n , изт $R_n, R_{n+1} := R_n \times$ изт **(MULTiPLY)**;

$$X \cdot Y = (X_{\text{МЛ}} + 2^8 \cdot X_{\text{СТ}}) \cdot (Y_{\text{МЛ}} + 2^8 \cdot Y_{\text{СТ}}) = \{\text{ДВ. ТОЧНОСТ}\}$$

$$= X_{\text{МЛ}} \cdot Y_{\text{МЛ}} + 2^8 \cdot (X_{\text{МЛ}} \cdot Y_{\text{СТ}} + X_{\text{СТ}} \cdot Y_{\text{МЛ}}) + 2^{16} \cdot X_{\text{СТ}} \cdot Y_{\text{СТ}}$$

DIV R_n , изт $R_{n+1} := R_n, R_{n+1} /$ изт **(ЧАСТНО)**

(DIVide) $R_n := R_n, R_{n+1} \bmod$ изт **(ОСТАТЪК)**.

ДЕСЕТИЧНИ ЧИСЛА

Има **няколко системи** за реализиране на работата с опаковани **ДКД числа**:

- ① **IBM-360**: задаване на **броя на цифрите** и **знака** в пълен **набор от аритметични МИ**.
- ② **6502**: два **режима** на работа на **АЛУ** – двоичен и десетичен, **за ADD и SUB**.
- ③ **180x86** и др.: набор от **МИ** за десетична **корекция след 8-битова двоична операция** – **DAA, DAS, DAM** и **DAD (Decimal Adjust)** при опаковани числа, **AAA, AAS, AAM** и **AAD (ASCII Adjust)** при неопаквани.

ПЛАВАЩА ЗАПЕТАЯ

В миналото (**IBM-360** и др.) добавката към ЦП (АЛУ и регистри) за **плаваща запетая се купува допълнително**.

При **първите МП** интеграцията позволява **АЛУ само с 8-битово** събиране и изваждане.

Ако се **оперира УУ на ЦП** на кристала **остава място** за реализиране на разширено АЛУ с ПЗ: +, -, ×, :, **e^x , $\ln x$, $\sin x$ и $\arctg x$** .

Тази ИС получава название **математически съпроцесор (копроцесор)**. **Днес той е част от кристала на ЦП.**

ЕДИН ОПЕРАНД

Тъй като при тях адресното поле за другия операнд (регистър) е свободно и обикновено всички операции имат еднакъв КОП, а това поле доуточнява действието им (икономия).

Увеличаване с 1 – **INC** пр пр:= пр + 1 (**INC**rement);

Намаляване с 1 – **DEC** пр пр:= пр – 1 (**DEC**rement);

Нулиране – **CLR** пр пр := 0 (**CL**eaR);

Допълнение до 1 – **COM** пр пр:=¬ пр (**COM**plement);

Допълнение до 2 – **NEG** пр пр:=¬ пр + 1 (**NEG**ate);

Проверка – **TST** пр пр ? 0 (**Te**ST).

INC/DEC И ADD/SUB #1

За какво е необходима **INC** пр? **ADD** пр,#1?

Двете МИ не са еднакви за **РУ**: **ADD** изменя бит **С**, а **INC** запазва стойността на този бит.

INC и **DEC** възникват в **PDP-11/20**

за управление на **цикли с преброяване**:

REPEAT

`cnt := cnt - 1;`

UNTIL cnt=0

LOOP: ...

`DEC CNT`

`BNE LOOP`

При операции с двойна точност **стойността на бит С** трябва да се пренесе **от края на едната итерация в следващата**.

В **PDP-11/45** се появява **SOB = DEC + BNE**.

ЛОГИЧЕСКИ ОПЕРАЦИИ

Аритметичните операции третират своите **операнди** като **единно цяло** ($01+01=10$).

Логическите операции третират операндите си като **масив от битове**: **всеки бит** се определя чрез **операция към съответните му**.

Основните логически операции са:

Конюнкция **AND** пр,изт пр := пр \wedge изт;

Дизюнкция **OR** пр,изт пр := пр \vee изт;

Сума по модул 2 **XOR** пр,изт пр := пр \oplus изт;

Проверка **BIT** пр,изт пр \wedge изт;

Нулиране (**PDP-11**) **BIC** пр,изт пр := пр \wedge \neg изт.

РАБОТА С ЕДИНИЧЕН БИТ

Използването на **логическите** операции с **подходящ непосредствен операнд** може да осигури **работа с единичен бит**.

Някои ЦП имат алтернативни **МИ** за работа с **единичен бит** чрез посочване на **номера му**:

Нулиране **BCLR** пр,***n*** пр[***n***] := **0**;

Включване **BSET** пр,***n*** пр[***n***] := **1**;

Промяна **BCHG** пр,***n*** пр[***n***] := \neg пр[***n***] (**CHanGe**);

Проверка **BTST** пр,***n*** **Z** := пр[***n***].

Номерът на бита може да се задава **статично** с **неп. операнд** или **динамично** в **регистър**.

ИЗМЕСТВАНИЯ

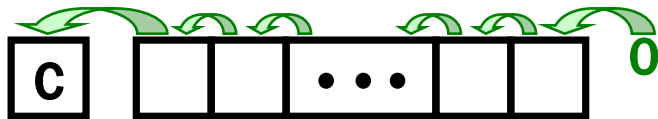
Логическите операции установяват правилно **Z** и **N**, нулират **V** и запазват **C**.

Основното им предназначение е опаковане и разопаковане на данни, за което трябва да се допълнят с **измествания** на битовете.

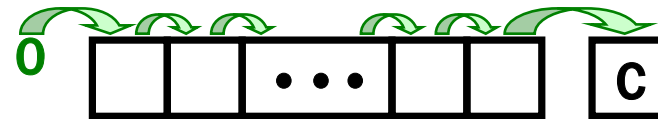
Изместванията (**Shift**) биват **в ляво** (**Left**) и **в дясно** (**Right**), **логически** (**Logical**) и **аритметически** (**Arithmetic**).

Обикновено изместването е на **1 бит**, но някои ЦП имат **МИ** със статично или **динамично задаване на броя** на битовете.

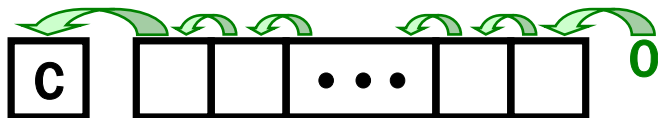
ПРИМЕР: ИЗМЕСТВАНЕ



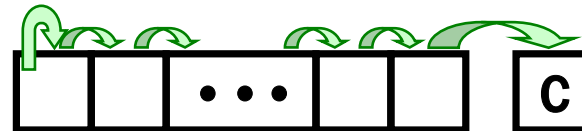
SLL (= $\times 2$ без знак)



SRL (= $/2$ без знак)



SLA (= $\times 2$ със знак)



SRA ($\approx /2$ със знак)

SLL установява $V=0$, а **SLA** – по промяната на знаковия бит до и след изместването.

В ляво \equiv умножение, **в дясно** \equiv деление.

SRA не е точно делене на 2 поради различие в окръгляването при двете МИ.

Логическо \equiv без знак, **аритметично** \equiv със.

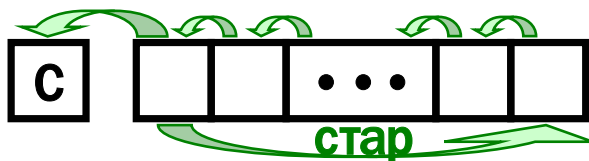
РОТАЦИИ

Ротациите (**Rotation**) са **разширение на изместванията** за увеличаване на размера.

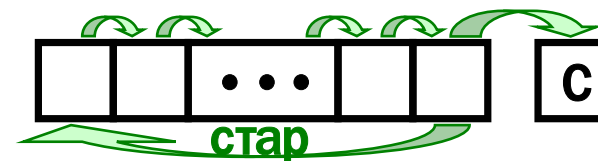
Те биват **със и без** участие на **бит C**.

Установяването на **бит V** в РУ **зависи** от проектанта: в **0** или **запазване**.

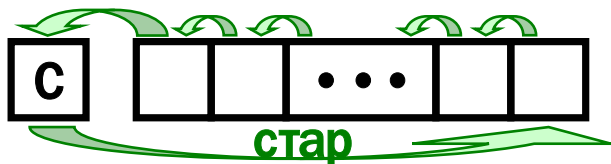
RL (в ляво **без C**)



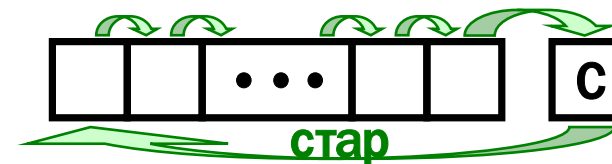
RR (в дясно **без C**)



RLC (в ляво **със C**)



RRC (в дясно **със C**)



УПРАВЛЯВАЩИ МИ

Те са **изключително важни**, защото са най-елегантният начин за използване на **условни действия** и **повторения** в програми на **МЕ**.

Целта на управляващите МИ е **да се наруши естественият ред за изпълнение**.

Затова се наричат и **инструкции за преход**.

Действието на тези МИ се състои в **замяна на ПБ** с изчисления от адресното поле **ЕА**.

Преходите не променят РУ и могат да бъдат **без** или **със** проверка на някакво **условие** в **РУ**, **без** или **със възможност за възврат**.

БЕЗУСЛОВЕН ПРЕХОД

Основната МИ за преход е без да се проверява каквото и да е условие (винаги):

JMP пр ПБ := EA(пр) (**JuMP**)

Старата стойност на ПБ се губи.

Безсмислени (и забранени) са регистровата адресация и непосредствен операнд.

Всички други обикновено са разрешени за да се осигури гъвкавост на разклоненията.

Запазването на ПБ осигурява възможност за възврат – МИ за преход към подпрограма:

CALL пр Запомни ПБ, ПБ := EA(пр)

ПРЕХОД КЪМ ПОДПРОГРАМА

Съществен елемент на тази МИ е **запазването на старата стойност на ПБ** за:

- 1 **възстановяване** на прекъснатия естествен ред на изпълнение (възврат);
- 2 **установяване на местоположението в ОП.**

Запазване на ПБ и възврат (RETurn):

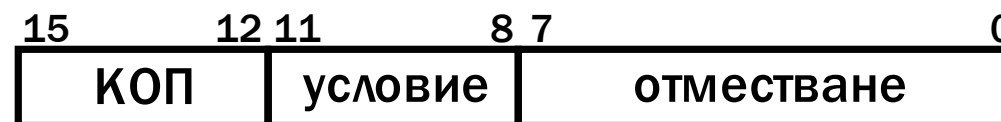
- 1 **IBM-360** в регистър (**BAL R,Адрес; BALR R,R₁**);
- 2 **PDP-8** в Адр и преход към **Адр+1 (JMS Адрес)**;
- 3 **днес в системния стек (Push(ПБ)); RET ПБ:=Pop.**

УСЛОВЕН ПРЕХОД

Тези МИ проверяват определено условие и **променят ПБ** само при наличието му.

Обикновено **адресът на прехода** се задава с **относителна адресация**, защото е **наблизо**.

Условието обикновено се кодира в **4-бита**, които при другите МИ кодират регистър.



При **монолитен РУ бит 1** посочва **стойност**, при която ще се извърши **преход**.



Задаване на условия в **IBM-360** (B и BR).

ПРОВЕРКА НА БИТОВЕ

При РУ с **независими битове** се кодират следните **16 проверки (Branch)**:

① **без проверка на условие:**

BRA винаги (**Always**) **BRN** никога (**Never**)

② **проверка на 1 бит:**

BCS при **C=1** (**Set**) **BCC** при **C=0** (**Clear**)

BVS при **V=1** **BVC** при **V=0**

BMI при **N=1** (**Minus**) **BPL** при **N=0** (**Plus**)

BEQ при **Z=1** (**Equal**) **BNE** при **Z=0** (**Not Eq**)

БИТОВЕ В РУ (прод.)

③ проверка на 2 бита:

BLT: $N \oplus V = 1$ (< със зН) **BGE**: $N \oplus V = 0$ (\geq със зН)

BLS: $C \vee Z = 1$ (\leq без зН) **BHI**: $C \vee Z = 0$ (> без зН)

④ проверка на 3 бита:

BLE при $(N \oplus V) \vee Z = 1$ (\leq със знак)

BGT при $(N \oplus V) \vee Z = 0$ (> със знак)

⑤ Често има и алтернативни названия:

BZ = **BEQ** $Z=1$ (Zero) **BNZ** = **BNE** $Z=0$ (Not Z)

BLO = **BCS** (< без зН) **BHS** = **BCC** (\geq без зН)

ДОПЪЛНИТЕЛНИ МИ

За по-голямо **удобство** при програмирането в групата на управляващите МИ често има и **допълнителни МИ, комбиниращи две** други:

PDP-11: SOB (Subtract One and Branch);
Z8000: DJNZ (Decrement and Jump if Not Zero);
I80x86: LOOP (брояч е CX или CL).

В съвременните ЦП се наблюдава **явна тенденция за** включване на допълнителни **специализирани МИ**, чиято **цел е да се улесни преводът** от ЕПВР на МЕ.

**БЛАГОДАРЯ ВИ
ЗА ВНИМАНИЕТО!**

**БЪДЕТЕ С МЕН И В
СЛЕДВАЩАТА ЛЕКЦИЯ,
КОЯТО ЩЕ НИ ОТВЕДЕ
В НЕВЕРОЯТНИЯ СВЯТ НА
ПОДПРОГРАМИТЕ
И ПАРАМЕТРИТЕ**