ՀՀ ԿՐԹՈՒԹՅԱՆ, ԳԻՏՈՒԹՅԱՆ, ՄՇԱԿՈՒՅԹԻ և ՍՊՈՐՏԻ ՆԱԽԱՐԱՐՈՒԹՅՈՒՆ

ՀԱՅԱՍՏԱՆԻ ԱԶԳԱՅԻՆ ՊՈԼԻՏԵԽՆԻԿԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ

(ՀԻՄՆԱԴՐԱՄ)

\_\_\_\_\_\_\_\_\_ ՏՀՏԷ \_ \_ինստիտուտ/ֆակուլտետ

Քոմփյութերային համակարգեր և ցանցեր ամբիոն

Ծրագրային ճարտարագիտություն մասնագիտություն

**ՀԱՇՎԵԲԱՑԱՏՐԱԳԻՐ**

Հաշվողական համակարգերի ծրագրային ապահովում և ճարտարապետություն\_\_\_\_\_

կուրսային աշխատանք

Թեմա՝ Տրված զանգվածի մեջ որոշել առավելագույնը և նշանը փոխել, արդյունքը դուրս բերել էկրանին։

ՏՏ119 \_\_ ակադեմիական խումբ

Ուսանող \_\_\_\_\_ Գևորգյան Վարուժան Սմբատի \_\_\_\_\_\_\_\_(ստորագրություն)

(ազգանուն, անուն հայրանուն)

Ղեկավար\_\_\_\_\_ Վիրաբյան Է.Վ. դասախոս \_\_\_\_\_\_\_(ստորագրություն) (ազգանուն, անուն հայրանուն)

Ամբիոնի վարիչ \_\_\_\_Կիրակոսյան Գ.Տ. տ.գ.դ., պրոֆ. \_\_ (ստորագրություն) (ազգանուն, անուն հայրանուն)

Երևան 2023

ՀՀ ԿՐԹՈՒԹՅԱՆ, ԳԻՏՈՒԹՅԱՆ, ՄՇԱԿՈՒՅԹԻ և ՍՊՈՐՏԻ ՆԱԽԱՐԱՐՈՒԹՅՈՒՆ

ՀԱՅԱՍՏԱՆԻ ԱԶԳԱՅԻՆ ՊՈԼՏԵԽՆԻԿԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ

(ՀԻՄՆԱԴՐԱՄ)

ՏՀՏԷ ինստիտուտ/ֆակուլտետ

Քոմփյութերային համակարգեր և ցանցեր ամբիոն

(ամբիոնի անվանումը)

Ծրագրային ճարտարագիտություն մասնագիտություն

(մասնագիտության անվանումը)

ՏՏ119 ակադեմիական խումբ

**ԿՈՒՐՍԱՅԻՆ ԱՇԽԱՏԱՆՔԻ**

**ԱՌԱՋԱԴՐԱՆՔ**

\_\_\_ Հաշվողական համակարգերի ծրագրային ապահովում և ճարտարապետություն\_\_\_\_

(առարկաի անվանումը)

Գևորգյան Վարուժան Սմբատի

(ուսանողի ազգանուն, անուն, հայրանուն)

1. Աշխատանքի թեման Տրված զանգվածի մեջ որոշել առավելագույնը և նշանը փոխել, արդյունքը դուրս բերել էկրանին։

2. Աշխատանքի նախնական տվյալները \_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

3.Հաշվեբացատրագրի բովանդակությունը

3.1 Ալգորիթմի\_բլոկ\_սխեմա

3.2 Ծրագրի\_կազմում\_ասեմբլեր\_լեզվով\_և ծրագրի\_կարգաբերում\_SASM փաթեթի միջոցով

4.Գրաֆիկական մասի ծավալը \_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 5.Կատարման ժամանակացույցը \_04.09.23թ.-22.12.23թ.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6.Աշխատանքի ղեկավար \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_Վիրաբյան Է.Վ.\_\_\_\_

(ստորագրություն) ( Ա.Ա.Հ.)

7. Ամբիոնի վարիչ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_Կիրակոսյան Գ.Տ. \_\_

(ստորագրություն) ( Ա.Ա.Հ.)

8.Ուսանող ստորագրություն՝ Ամսաթիվ

**Խնդրի լուծման ծրագիրը C++ ծրագրավորման լեզվով ՝**

#include <iostream>

#include <cstdlib>

#include <ctime>

#include <cmath>

float roundToDecimalPlaces(float number, int decimalPlaces) {

float multiplier = std::pow(10.0f, static\_cast<float>(decimalPlaces));

return std::round(number \* multiplier) / multiplier;

}

int main(){

int size = 10;

float\* ptr = new float[size];

// Seed the random number generator with the current time

std::srand(static\_cast<unsigned int>(std::time(nullptr)));

// Define the range for the random float

float min = -100.0f;

float max = 100.0f;

int decimalPlaces = 2;

std::cout << "Initial list:\n";

for(int i=0; i<size; i++){

float random\_float = min + static\_cast<float>(std::rand()) / (static\_cast<float>(RAND\_MAX) / (max - min));

ptr[i] = roundToDecimalPlaces(random\_float, decimalPlaces);

std::cout << ptr[i] << " ";

}

max = min;

for(int i=0; i<size; i++){

if (max < ptr[i]){

max = ptr[i];

}

}

std::cout << "\nMax value is: " << -max;

delete[] ptr;

return 0;

}

**Խնդրի լուծման ալգորիթմը Assembler-ով**

%include "io.inc"

section .data

array dd 12, 23, 2, 52, 9

length dd 5

max dd 0

count dd 0

section .text

global main

main:

mov ebp, esp

xor eax, eax

xor ebx, ebx

mov ecx, [length]

xor esi, esi

lea ebx, [array]

mov eax,[ebx, esi\*4]

loop: inc esi

dec ecx

jz stop

cmp eax,[ebx, esi\*4]

jg loop

mov eax,[ebx, esi\*4]

jmp loop

stop:

PRINT\_STRING "Max value is: "

PRINT\_DEC 4, eax

mov [max], eax

neg eax

NEWLINE

PRINT\_STRING "Negative value of maximum: "

PRINT\_DEC 4, eax

xor ebx, ebx

;mov eax, -1 ;for testing code

bsr ecx, eax

add ecx, 2

loop1:

dec ecx

jz stop1

rcr eax, 1

jc plus

jmp loop1

plus:

add ebx, 1

jmp loop1

stop1:

mov [count], ebx

NEWLINE

PRINT\_STRING "Count of 1 is: "

PRINT\_DEC 4, count

xor ecx, ecx

bsr ecx, res

NEWLINE

PRINT\_STRING "BSR: "

PRINT\_DEC 4, ecx

NEWLINE

xor ecx, ecx

bsf ecx, res

PRINT\_STRING "BSR: "

PRINT\_DEC 4, ecx

NEWLINE

PRINT\_HEX 4, res

rol res, 3

NEWLINE

PRINT\_STRING "BSR: "

PRINT\_DEC 4, res

PRINT\_HEX 4, res

ret

**Ներածություն**

Ասեմբլեր լեզուն ցածր մակարդակի լեզու է ԷՀՄ միկրոպրոցեսորների, միկրովերահսկիչ սարքերի եւ ծրագրավորվող այլ սարքերի համար։ Այն մեքենայական հրամանի սիմվոլիկ ներկայացումն է եւ հատկորոշվում է ԷՀՄ ճարտարապետությամբ։

Ասեմբլեր անվանում են նաեւ ասեմբլերի թարգմանչին։

Բանավեճը՝ ասեմբլերի կիրառության օգտակարության վերաբերյալ, համեմատած բարձր մակարդակի լեզուների հետ, ներկայումս ևս շարունակվում է։ Ասեմբլեր լեզուն նախընտրում են օգտագործել հետեւյալ իրավիճակներում.

* Ծրագիրն աշխատում է անմիջապես սարքերի հետ (օրինակ՝ դրայվերները, ընդհատումը մշակող ծրագրերը):

•  Անհրաժեշտ է օգտագործել պրոցեսորի հրամաններ, որոնք հասանելի չեն բարձր մակարդակի լեզուների թարգմանիչներին (compiler)։ Օրինակ՝ պտույտի հրամանները, որոնք օգտագործվում են կոդավորման ծրագրերում:

•   Օպտիմիզացումն անհրաժեշտություն է (օրինակ՝ խաղերի ծրագրերում):

•   Գոյություն չունի բարձր մակարդակի լեզու նոր կամ հատուկ պրոցեսորների համար:

•   Անհրաժեշտ է փոփոխել գոյություն ունեցող երկուական կոդը, երբ նրա սկզբնաղբյուր բարձր մակարդակի լեզվով գրված ծրագիրը չկա:

•   Անհրաժեշտ է գրել կոմպիլյատոր (compiler), որը գեներացնելու է ասեմբլեր կոդ (նման աշխատանքի համար պահանջվում է ասեմբլերի լավ իմացություն):

x86 ասեմբլեր լեզուն ունի երկու հիմնական շարահյուսություն` Ինթել շարահյուսություն եւ AT&T շարահյուսություն։ Ինթել շարահյուսությունը օգտագործվում է MS-DOS եւ Windows ՕՀ համար, իսկ AT&T շարահյուսությունը՝ Unix ՕՀ համար։ Արտաքին տեսքով այս երկու շարահյուսությունները տարբերվում են միմյանցից (օրինակ՝ 37 ըստ Ինթել շարահյուսության գրված ‘MOV AX,1’ հրամանը համարժեք է ըստ AT&T շարահյուսության գրված ‘MOVL $1, %EAX’)։ Մենք կծանոթանանք Ինթել շարահյուսությանը։

Ասեմբլեր լեզուն մեծատառ-փոքրատառերի նկատմամբ ոչ զգայուն (case-insensitive) լեզու է։

**Օպերանդների հասցեավորման եղանակները**

IA-32 օպերանդները կարող են գտնվել.

* •  ռեգիստրում,
* •  հիշողության մեջ,
* •  անմիջապես հրամանում (անմիջական օպերանդ),
* •  մուտքի/ելքի կայանում:

Գործողության արդյունքը կարող է գրանցվել.

* + •  ռեգիստրում,
  + •  հիշողության մեջ,
  + •  մուտք/ելքի կայանում:

**Հասցեավորման եղանակ (Addressing mode)** ասելով՝ կհասկա- նանք օպերանդի տրման եղանակը։ IA-32 կան հասցեավորման հետեւյալ եղանակները:

**Ռեգիստրային հասցեավորում**

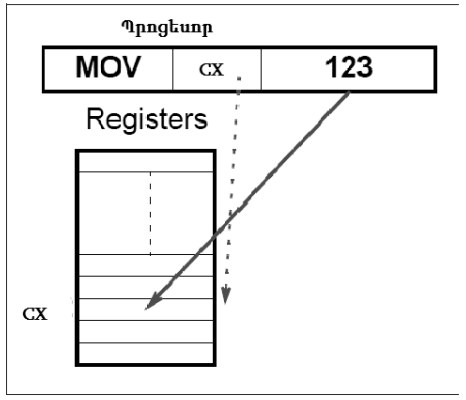
Այս հասցեավորման դեպքում օպերանդները ռեգիստրներ են:

MOV AX, BX ; ռեգիստրային հասցեավորում

**Անմիջական հասցեավորում**

Անմիջական հասցեավորում օգտագործվում է երկրորդ օպե- րանդի համար։ Այն ներկայացնում է անմիջական արժեք, որը գրանցվում է մեքենայական հրամանում։

MOV CX, 123



Եթե օպերանդը գտնվում է հիշողության մեջ, ապա այն կարող է տրվել **ուղղակի կամ ոչ ուղղակի**։ Արդյունավետ հասցեի (Effective Address/EA) կամ շեղման (Offset) ձեւավորմանը մասնակցում են հետեւյալ բաղադրիչները.

•  բազա (base),

•  ինդեքս (Index),

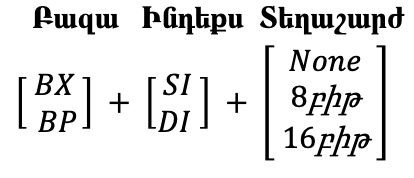
•  մասշտաբ (Scale),

•  տեղաշարժ (Displacement)։

Հիշեցնենք, որ հիշողության սեգմենտային կազմակերպման ժամանակ հասցեն տրվում է սեգմենտ/ընտրիչ (selector) եւ շեղում (կամ արդյունավետ հասցե) զույգի միջոցով։

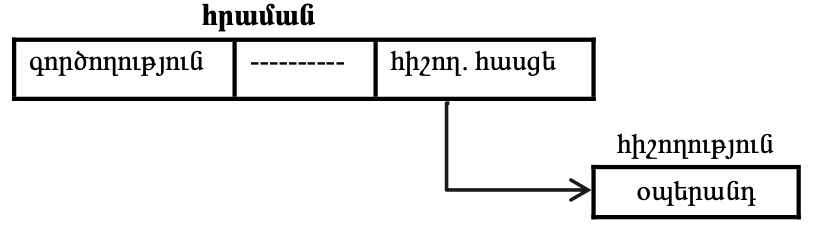
16-բիթանոց հասցեավորման դեպքում արդյունավետ հասցեն = բազա + ինդեքս + տեղաշարժ.

**Բազա Ինդեքս Տեղաշարժ**

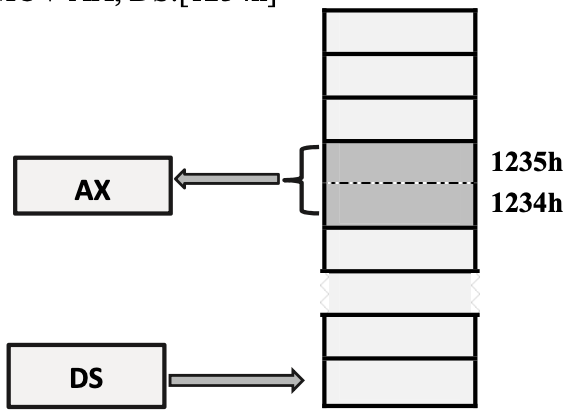


**Ուղղակի հասցեավորում**

Արդյունավետ հասցեի ձեւավորմանը մասնակցում է միայն տեղա- շարժը (displacement), որը գրանցվում է մեքենայական հրամանում:



Օրինակ ՝ MOV AX, DS: [1234h]

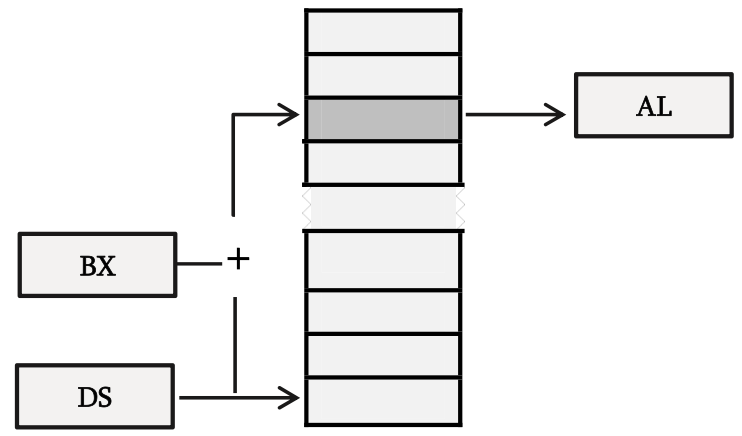


**Անուղղակի ռեգիստրային հասցեավորում**

Այս հասցեավորման դեպքում արդյունավետ հասցեի ձեւավոր- մանը մասնակցում է միայն բազան կամ ինդեքսը:

Օրինակ

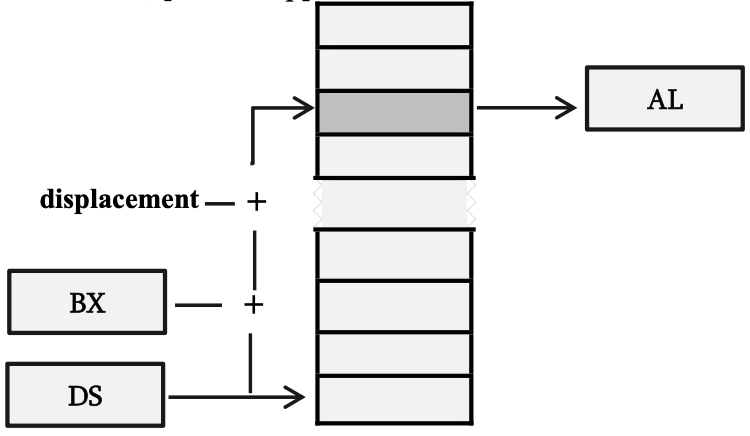
MOV AL, [BX]



**«Բազա + տեղաշարժ» անուղղակի հասցեավորում**

Այս հասցեավորման դեպքում արդյունավետ հասցեի ձեւավոր- մանը մասնակցում են բազան կամ ինդեքսը եւ տեղաշարժը (displacement):

Օրինակ՝ MOV AL, [BX + disp]



**«(Ինդեքս \* մասշտաբ) + տեղաշարժ» անուղղակի հասցեավորում**

Այս հասցեավորման դեպքում արդյունավետ հասցեից մասնակ- ցում են կա՛մ ինդեքսային ռեգիստրը, կա՛մ ինդեքսային ռեգիստրն ու տեղաշարժը:

Մասշտաբն օգտագործվում է 386-երից սկսած։

Օրինակ ՝ MOV EAX, A[EBX \* 4] ; որտեղ A DD 6, 89,...  
Այս հասցեավորման եղանակն առաջարկում է արդյունավետ եղանակ` ստատիկ զանգվածի տարրերի հետ աշխատելու համար, երբ տարրի չափը 2, 4 կամ 8 բայթ է։ Տեղաշարժը տեղայնացնում է զանգվածի սկիզբը, իսկ ինդեքսային ռեգիստրը ցույց է տալիս զանգվածի տարրի ինդեքսը։

**«Բազա + ինդեքս + տեղաշարժ» անուղղակի հասցեավորում**

Օրինակ ՝

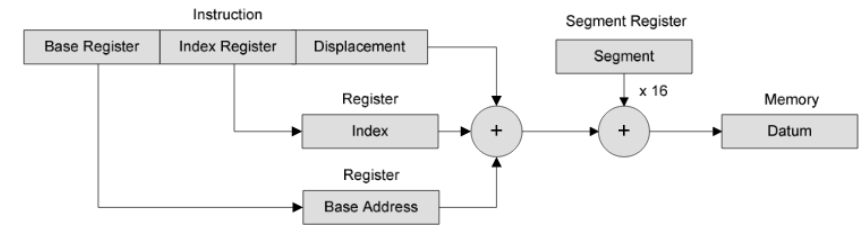
MOV DX, [BX + DI]

MOV DX, [DI][BX]

MOV DX, [BX][DI]

MOV DX, A[BX][SI]

Առաջին 3 օրինակներում տեղաշարժը բացակայում է։



Երկու ռեգիստրների միաժամանակյա օգտագործումը հնարա- վորություն է տալիս աշխատելու կա՛մ երկչափ զանգվածի (տեղա- շարժը զանգվածի սկզբի հասցեն է), կա՛մ գրառումների զանգվածի (տեղաշարժը գրառման դաշտի շեղումն է) հետ։

**«Բազա + ինդեքս \* մասշտաբ + տեղաշարժ» (բազաինդեքսային հասցեավորում մասշտաբով) անուղղակի հասցեավորում**

Օրինակ ՝

MOV AL, 20[EBX \* 2 + ECX] ; գրության MOV AL,[20+EBX\*2+ECX] ;երեքձեւերը MOV AL, [EBX \* 2 + ECX + 20] ; համարժեք են։

Հասցեավորման բոլոր բաղադրիչների օգտագործումն արդյու- նավետ է երկչափ զանգվածի տարրերին դիմելու համար, երբ տարրի չափը 2, 4 կամ 8 բայթ է։

**Բիթային որոնում**

**BSF** – **Բիթային որոնում առաջ / Bit Scan Forward (386+)**

Շարահյուսություն: **BSF ընդունիչ, աղբյուր**

Նկարագրություն: ընդունիչ BSF(աղբյուր)

Դրոշներ: AF = ? CF = ? OF = ? PF = ? SF = ? ZF

Ընդունիչը կարող է լինել ռեգիստր (reg16/reg32), իսկ աղբյուրը՝ ռեգիստր (reg16/reg32) կամ հիշողության հասցե (mem16/mem32)։ Օպերանդների չափերը պետք է համընկնեն: BSF հրամանը որոնում է աղբյուրի մեջ կրտսեր՝ 0 բիթից սկսած առաջին 1 արժեք ունեցող բիթը եւ նրա համարը (0-ական բիթից հաշված) գրում ընդունիչի մեջ: Եթե աղբյուրի արժեքը 0 է, ապա ընդունիչի արժեքն անորոշ է: AF, CF, OF, PF, SF դրոշների արժեքներն անորոշ են: ZF-ը 1 է, եթե աղբյուրի արժեքը 0 է, հակառակ դեպքում՝ ZF = 0։

**BSR** – **Բիթային որոնում ետ / Bit Scan Reverse (386+)**

Շարահյուսություն: **BSR ընդունիչ, աղբյուր**

Նկարագրություն: ընդունիչ BSR(աղբյուր)

Դրոշներ: AF = ? CF = ? OF = ? PF = ? SF = ? ZF

Ընդունիչը կարող է լինել ռեգիստր (reg16/reg32), իսկ աղբյուրը՝ ռեգիստր (reg16/reg32) կամ հիշողության հասցե (mem16/mem32)։ Օպերանդների չափերը պետք է համընկնեն:

BSF հրամանը որոնում է աղբյուրի մեջ ավագ՝ 15 կամ 31 բիթից սկսած առաջին 1 արժեք ունեցող բիթը եւ նրա համարը (0-ական բի- թից հաշված) գրում ընդունիչի մեջ: Եթե աղբյուրի արժեքը 0 է, ընդու-նիչի արժեքն անորոշ է: AF, CF, OF, PF, SF դրոշների արժեքներն անորոշ են: ZF-ը 1 է, եթե աղբյուրի արժեքը 0 է, հակառակ դեպքում՝ ZF = 0։

Օրինակ ՝

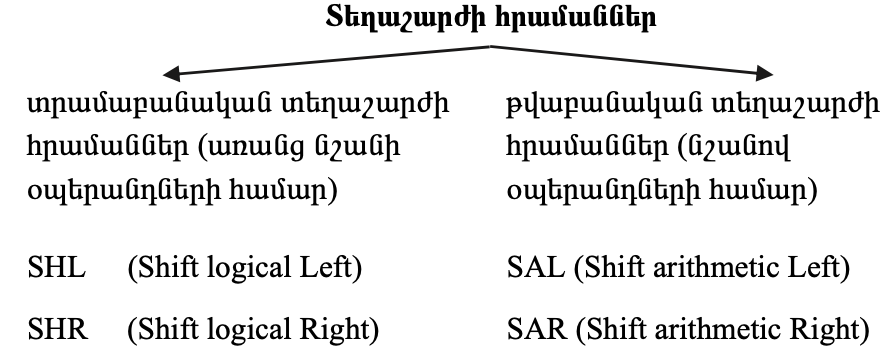
MOV CX, 1256h

BSF AX, CX ; AX=1

BSR DX, CX ; DX=12

**Տեղաշարժի հրամաններ**

Տեղաշարժի հրամանները լինում են տրամաբանական եւ թվա- բանական եւ կատարում են տեղաշարժ դեպի աջ կամ ձախ:

****

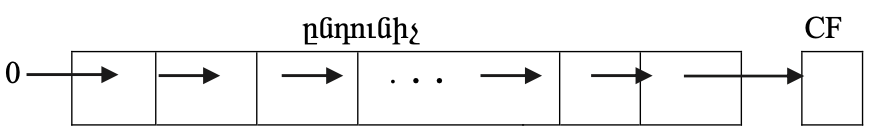
Տրամաբանական տեղաշարժի ժամանակ ազատված բիթերում գրվում են 0-ներ: Թվաբանական աջ տեղաշարժի ժամանակ ազատ- ված բիթերում ընդունիչի նշանային բիթի արժեքն է կրկնվում, իսկ ձախ տեղաշարժի ժամանակ ազատված բիթերը լրացվում են 0–նե- րով:

**SHR** – **Տրամաբանական աջ տեղաշարժ /Shift logical Right**

Շարահյուսություն: **SHR ընդունիչ, հաշվիչ**

Նկարագրություն: ընդունիչ SHR( ընդունիչ )

Դրոշներ: AF=? CF OF PF SF ZF



Օրինակ`

MOV AL, 43 ; AL = 0010101**1**B  
SHR AL, 1 ; AL = **0**0010101B

Դրոշները՝ CF = 1, OF = 0, SF = 0, ZF = 0, PF = 0, AF = ?

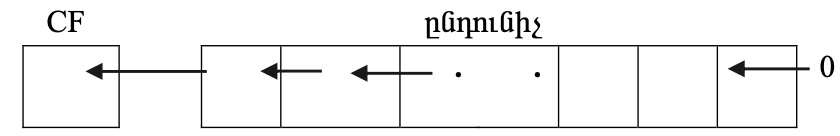
SHR հրամանը կարելի է օգտագործել (եւ արժի օգտագործել, քանի որ այն ավելի արագ կատարվող հրաման է) ընդունիչի արժե- քը որպես առանց նշանի թիվ 2-ի հաշվիչ աստիճանի վրա բաժանելու համար: Օրինակում AL ռեգիստրում գրվել էր 43, հրամանից հետո AL-ի արժեքը հավասար է 21, այսինքն՝ 43/2: Եթե կրկին կատարվի SHR AL, 1 հրամանը, կամ միանգամից կատարվի 2 բիթ տեղաշարժ դեպի աջ (MOV CL, 2 եւ SHR AL, CL), ապա AL ռեգիստրի արժեքը կլինի 43/4 = 10:

**SHL/SAL** – **Տրամաբանական / թվաբանական ձախ տեղաշարժ/ Shift logical / arithmetic Left**

Շարահյուսություն: **SHL ընդունիչ, հաշվիչ, SAL ընդունիչ, հաշվիչ**

Նկարագրություն: ընդունիչ SHL( ընդունիչ )

Դրոշներ: AF = ? CF OF PF SF ZF



SHL եւ SAL հրամանները նույնն են (ունեն նույն գործողության կոդը):

Օրինակ`

MOV AL, 43 ; AL = 00101011B  
SHL AL, 1 ; AL = 01010110B  
Դրոշները՝ CF = 0, OF = 0, SF = 0, ZF = 0, PF = 1, AF = ?

SHL AL, 1 հրամանը համարժեք է SAL AL, 1 հրամանին։

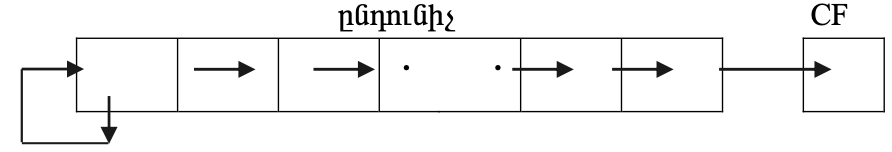
Հրամանը կարելի է օգտագործել (եւ արժի օգտագործել, քանի որ այն ավելի արագ կատարվող հրաման է) ընդունիչի արժեքը որ- պես առանց նշանի թիվ 2-ի հաշվիչ աստիճանով բազմապատկելու համար: Օրինակում AL ռեգիստրում գրվել էր 43, հրամանից հետո AL-ի արժեքը հավասար է 86, այսինքն` 43 \* 2: Եթե մեկ անգամ եւս կատարվի SHL AL,1 հրամանը, կամ կատարվի միանգամից 2 բիթ տեղաշարժ դեպի ձախ (MOV CL, 2 եւ SHL AL, CL), ապա AL ռեգիստրի արժեքը կլինի 43 \* 4 = 172:

**SAR** – **Թվաբանական աջ տեղաշարժ / Shift arithmetic Right**

Շարահյուսություն: **SAR ընդունիչ, հաշվիչ**

Նկարագրություն: ընդունիչ SAR ( ընդունիչ )

Դրոշներ: AF = ? CF OF PF SF ZF



Օրինակ `

MOV AL, 43 ; AL = **0**010101**1**B  
SAR AL, 1 ; AL = **00**010101B  
Դրոշները՝ CF = 1, OF = 0, SF = 0, ZF = 0, PF = 0, AF = ?

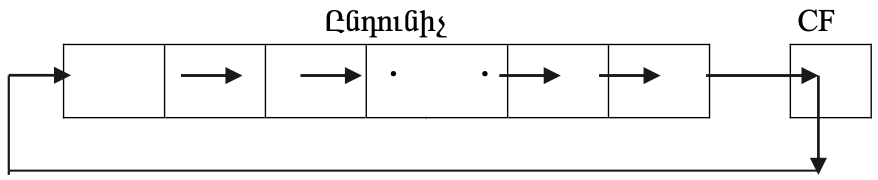
Հրամանը կարելի է օգտագործել (եւ արժի օգտագործել, քանի որ այն ավելի արագ կատարվող հրաման է) ընդունիչի արժեքը որ- պես նշանով թիվ 2-ի հաշվիչ աստիճանի վրա բաժանելու համար: Օրինակում AL ռեգիստրում գրվել էր 43, հրամանից հետո AL–ի արժեքը հավասար է 21, այսինքն՝ 43/2: Եթե մեկ անգամ եւս կատարվի SAR AL, 1 հրամանը, կամ կատարվի միանգամից 2 բիթ տեղաշարժ (MOV CL, 2 եւ SAR AL, CL), ապա AL ռեգիստրի արժեքը կլինի 43/4 = 10:

**RCR** – **Ցիկլիկ աջ տեղաշարժ փոխանցման դրոշով / Rotate through Carry Right**

Շարահյուսություն: **RCR ընդունիչ, հաշվիչ**

Նկարագրություն: ընդունիչ RCR (ընդունիչ)

Դրոշներ: CF, OF



Օրինակ ՝

CLC ; CF = **0**

MOV AL, 43 ; AL = 0010101**1**B

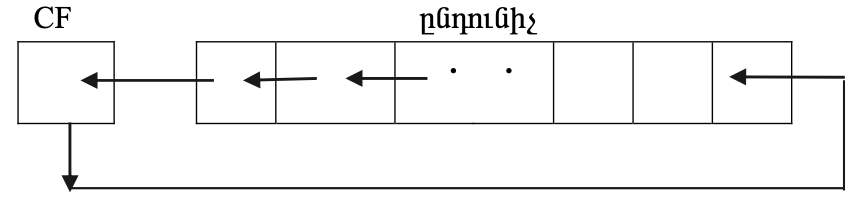
RCR AL, 1 ; AL = **0**0010101B, դրոշները՝ CF = **1**, OF = 0

**RCL** – **Ցիկլիկ ձախ տեղաշարժ փոխանցման դրոշով / Rotate through Carry Left**

Շարահյուսություն: **RCL ընդունիչ, հաշվիչ**

Նկարագրություն: ընդունիչ RCL( ընդունիչ )

Դրոշներ: CF OF



Օրինակ `

CLC ; CF = 0  
MOV AL, 43 ; AL = **0**0101011B

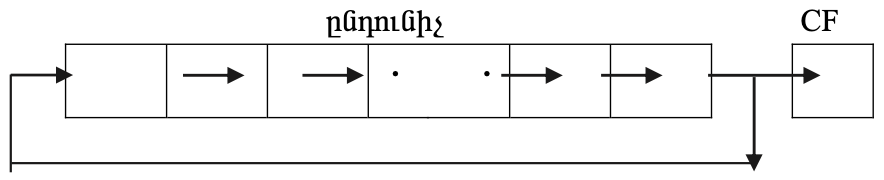
RCL AL, 1 ; AL = 0101011**0**B, դրոշները՝ CF = **0**, OF = 0

**ROR** – **Ցիկլիկ աջ տեղաշարժ / Rotate right**

Շարահյուսություն: **ROR ընդունիչ, հաշվիչ**

Նկարագրություն: ընդունիչ ROR( ընդունիչ )

Դրոշներ: CF OF



Օրինակ`

CLC ; CF = 0  
MOV AL, 43 ; AL = 0010101**1**B

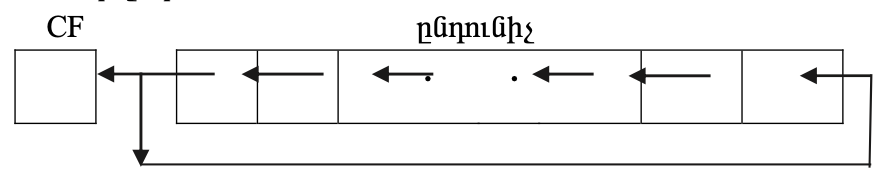
ROR AL, 1 ; AL = **1**0010101B, դրոշները՝ CF = **1**, OF = 1

**ROL** – **Ցիկլիկ ձախ տեղաշարժ / Rotate left**

Շարահյուսություն: **ROL ընդունիչ, հաշվիչ**

Նկարագրություն: ընդունիչ ROL( ընդունիչ ) CF OF

Դրոշներ: CF



Օրինակ `

CLC ; CF = 0  
MOV AL, 43 ; AL = **0**0101011B

ROL AL, 1 ; AL = 0101011**0**B, դրոշները՝ CF = **0**, OF = 0

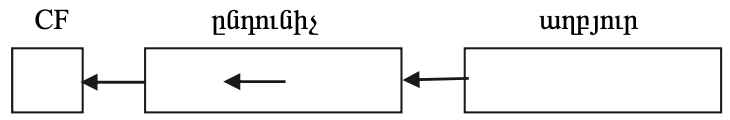
**SHLD** – **Տրամաբանական կրկնակի ձախ տեղաշարժ (386+)/ Shift Left Double**

Շարահյուսություն: **SHLD ընդունիչ, աղբյուր,հաշվիչ**

Նկարագրություն: ընդունիչ SHLD (ընդունիչ,աղբյուր)

Դրոշներ: AF = ? CF OF PF SF ZF

Ընդունիչը կարող է լինել ռեգիստր (reg16/reg32) կամ հիշողու- թյան հասցե (mem16/mem32), աղբյուրը կարող է լինել ընդունիչի տի- պին համապատասխան ռեգիստր (reg16/reg32), իսկ հաշվիչը՝ թիվ կամ CL ռեգիստրը: Դրոշները ստանում են հետեւյալ արժեքները. AF- ը անորոշ է, CF-ում ընդունիչից դուրս եկած վերջին բիթի արժեքն է, OF-ի արժեքը 1-ից ավելի տեղաշարժերի դեպքում անորոշ է, իսկ 1 դիրք տեղաշարժի ժամանակ OF-ը հավասար է 1-ի, եթե տեղաշար- ժից հետո ընդունիչի նշանը փոխվել է, եւ 0՝ հակառակ դեպքում: PF, SF, ZF-ը որոշվում են ըստ արդյունքի։



Օրինակ`

MOV AX, 43 ;AX= 0000 0000 00101011B

MOV BX, 9300h ;BX= 10010011 00000000B

SHLD AX, BX, 2 ;AX= 00 0000 0010101110B   
; BX-ն անփոփոխ է, դրոշները՝ CF = 0,

; OF = ?, SF = 0, ZF = 0, PF = 0, AF = ?

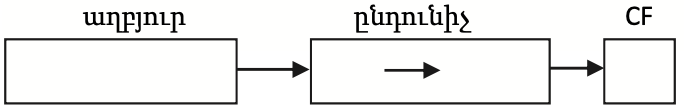
**SHRD** – **Տրամաբանական կրկնակի աջ տեղաշարժ (386+) / Shift Right Double**

Շարահյուսություն: **SHRD ընդունիչ, աղբյուր, հաշվիչ**

Նկարագրություն: ընդունիչ SHRD( ընդունիչ, աղբյուր)

Դրոշներ: AF = ? CF OF PF SF ZF

Ընդունիչը կարող է լինել ռեգիստր (reg16/reg32) կամ հիշողու- թյան հասցե (mem16/mem32), աղբյուրը կարող է լինել ռեգիստր (reg16/reg32), իսկ հաշվիչը՝ թիվ կամ CL ռեգիստր: Դրոշները ստա- նում են հետեւյալ արժեքները. CF-ում ընդունիչից դուրս եկած վերջին բիթի արժեքն է, OF-ը 1 բիթ տեղաշարժի դեպքում ընդունում է 1 արժեք, եթե տեղի է ունեցել նշանի բիթի փոփոխություն, հակառակ դեպքում ընդունում է 0 արժեք, 1-ից ավելի բիթերի տեղաշարժի դեպ- քում OF դրոշն անորոշ է, AF-ը անորոշ է, PF, SF, ZF-ը որոշվում են ըստ արդյունքի արժեքի։



Օրինակ`

MOV AX, 43 ; AX = 0000 000000101011B

MOV BX, 0093h ; BX = 0000000010010011B

SHRD AX, BX, 2 ; AX = 1100 0000 0000 1010B,  
; BX-ն անփոփոխ է, դրոշները՝ CF = 1,

; OF = ?, SF = 1, ZF = 0, PF = 1, AF = ?