Set de instructiuni TMS320C2x

ABS-valoarea absolută a acumulatorului

Sintaxa : [etichetă] ABS [;com]
Operanzi : nici unul
Cuvinte : 1
Execuţie : (PC)+1→PC
|(ACC)|→ACC
Influenţează OV ; este influenţată de OVM.
C=0;
Nu este influenţată de SXM.

Descriere: Dacă conţinutul acumulatorului este mai mare sau egal cu zero, el rămâne neschimbat de execuţia instruncţiunii ABS . Dacă conţinutul acumulatorului este mai mic decât zero, el va fi înlocuit cu valoarea lui în complement faţă de 2.Valoarea 80000000h este un caz special. Dacă nu este setat modul depaşire, ABS 80000000h este 80000000h. În modul depaşire, ABS de 80000000h este 7FFFFFFFh. În ambele cazuri, bitul de stare OV este setat. Bitul de transport (C) la C2x este întotdeauna resetat la zero de execuţia lui ABS.

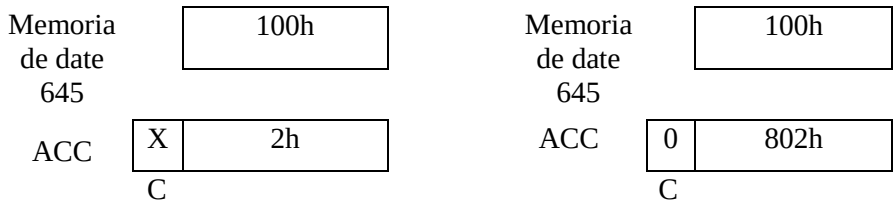
Exemplu: ABS

	Inainte de execuţie			După execuţie	
ACC	X	7EF1564h	ACC	0	7EF1564h
	C			C	
ACC	X	0FFFFFFFEh	ACC	0	2h
	C			C	

ADD – adună la acumulator cu deplasare

Sintaxa: ad.dir.: [etichetă] ADD dma[,deplasare] [;com]
ad.ind.: [etichetă] ADD {ind}[.deplasare[,ARP următor]] [;com]
Operanzi: 0≤dma≤127
0≤ARP următor ≤7
0≤deplasare≤15(implicit 0)
Cuvinte : 1
Execuţie: (PC)+1→PC
(ACC)+[(dma)x$2^{deplasare}$]→ACC
Dacă SXM=1, atunci (dma) va fi cu semn extins.
Dacă SXM=0, atunci (dma) nu va fi cu semn extins.
Influenţează OV; este influenţată de OVM si SXM.
Influenţează C.
Descriere: Conţinutul locaţiei de memorie adresate este deplasat la stânga si adunat la acumulator. În tmpul deplasării, biţii inferiori se completează cu zero. Biţii superiori sunt completaţi cu valoarea bitului de semn dacă SXM=1 şi cu zero dacă SXM=0. Rezultatul este memorat în acumulator.

Exemplu: ADD 5,3 ;(DP=5)
Sau
ADD *,3 ;dacă registrul auxiliar curent conţine 645



ADDH-adună la cuvântul superior al acumulatorului

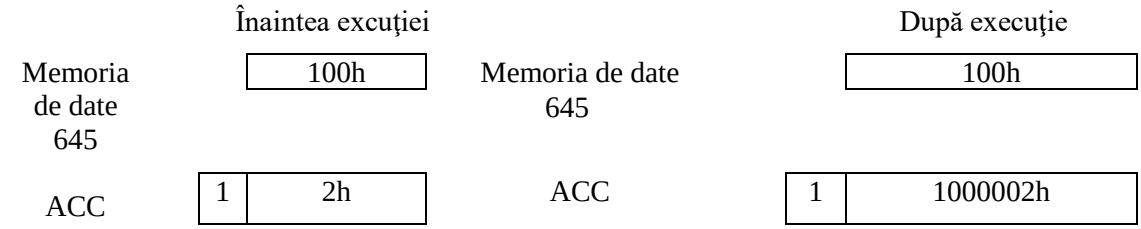
Sintaxa: ad.dir.: [etichetă] ADDH dma [;com]
ad. ind.: [etichetă] ADDH {ind}[,ARP următor] [;com]
Operanzi: 0≤dma≤127
0≤ARP următor ≤7
Cuvinte: 1
Execuţie: (PC)+1→PC
(ACC) +[(dma)x 2^{16}] →ACC
Influenţează OV ; este influenţată de OVM.
Influenţează C.
Biţii inferiori ai acumulatorului nu sunt influenţaţi.

Descriere : Conţinutul locaţiei de memorie de date adresate este adunat jumătăţii superioare a acumulatorului (biţii 31-16). Biţii inferiori nu sunt influenţaţi de ADDH. Bitul C este setat dacă rezultatul adunării generează transport; altfel, C rămâne nemodificat. Bitul carrz poate fi doar setat si nu resetat de instrucţiunea ADDH.

Exemplu : ADDH 5,3 ;(DP=5)

sau

ADDH *,3 ;dacă registrul auxiliar curent conţine 645h



ADDK – adună la acumulator un octet, imediat

Sintaxa: [etichetă] ADDK constantă [;com]
Operand: 0≤constantă≤255
Cuvinte: 1
Execuţie: (PC)+1→PC
(ACC) + constantă pozitivă de 8 biţi →ACC
Influenţează OV şi C; este influenţată de OVM;
Nu este influenţată de SXM.

Descriere: Valoarea imediată pe 8 biţi, aliniată la dreapta, este adunată la acumulator şi rezultatul înlocuieşte conţinutul acumulatorului. Valoarea imediată este tratată ca un număr pozitiv pe 8 biţi, indiferent de valoarea lui SXM.

Exemplu: ADDK 0Dh

		Înaintea execuţiei				După execuţie
ACC	X	2195180h		ACC	0	219518Dh
	C				C	

ADLK-adună la acumulator un cuvânt, imediat, cu deplasare

Sintaxa : [etichetă] ADLK constantă [;com]
Operanzi: constantă de 16 biţi
0≤deplasare≤15(implicit=0)
Cuvinte : 2
Execuţie : (PC)+2→PC
(ACC) + [constantă x $2^{deplasare}$] →ACC
Dacă SXM=1, atunci -32768≤constanta≤32767
Dacă SXM=1, atunci 0≤constanta≤65535
Influenţează OV; este influenţată de OVM şi SXM.
Influenţeză C.
Descriere : Valoarea imediată de 16 biţi, deplasată la stânga conform specificaţiei, este adunată la accumulator. Rezultatul înlocuieşte conţinutul acumulatorului. SXM determina dacă constanta este tratată ca un număr cu semn în complement faţă de 2 sau ca un număr fără semn. Numărătorul de deplasare este optional şi, când lipseşte, are implicit valoarea zero.

Exemplu : ADLK 11h,4

		Înaintea execuţiei				După execuţie
ACC	X	219Ch		ACC	0	22ACh
	C				C	

AND – Şi cu acumulator

Sintaxa : ad.dir.: [etichetă] AND dma [;com]
ad.ind.:[etichetă] AND {ind}[,ARP următor] [;com]
Opreanzi: 0≤dma≤127
0≤ARP următor≤7
Cuvinte: 1
Execuţie: (PC)+1→PC
(ACC(15-0)) ŞI (dma) →ACC(15-0)
0→ACC(31-16)
Nu este influenţată de SXM.
Descriere: Jumătatea inferioară a acumulatorului este supusă unei operaţii ŞI cu conţinutul locaţiei de memorie de date adresată. Jumătatea superioară a acumulatorului este supusă unei operaţii ŞI cu zero. De aceea, jumătatea superioară a acumulatorului este întotdeauna zero după execuţia unei instruncţiuni AND.

Exemplu: AND 5 ;(DP=5)
Sau
AND * ; dacă registrul auxiliar curent conţine 645

		Înaintea execuţiei				După execuţie
Memoria de date 645		0F100h		Memoria de date 645		0F100h
ACC	X	45E5F22h		ACC	0	00005100h
	C				C	

ANDK- ŞI imediat cu acumulator, cu deplasare

Sintaxa : [etichetă] ANDK constantă[,deplasare] [;com]
Operanzi :constantă de 16 biţi
0≤deplasare≤15(implicit = 0)
Cuvinte : 2
Execuţie: (PC)+1→PC
(ACC(30-0)) ŞI [constantă x $2^{deplasare}$] →ACC(30-0)
0→ACC(31) şi toţi biţii neocupaţi de constanta deplasată
Nu este influenţată de SXM.
Descriere: Constanta imediată de 16 biţi este deplasată la stânga şi supusă unei operaţii ŞI cu acumulatorul. Rezultatul este depus în acumulator. Biţii inferiori de sub şi biţii superiori de deasupra valorii deplasate sunt trataţi ca zerouri şi anulează, astfel, biţii corespunzători ai acumulatorului. Este de remarcat că cel mai semnificativ bit al acumulatorului este întotdeauna zero, indiferent de valoarea deplasării.
Exemplu: ANDK 0FFFFh, 12

		Înaintea execuţiei			După execuţie
ACC	X	123456789h	ACC	X	02345000h
	C			C	

APAC- adunăregistrul P la acumulator

Sintaxa : [etichetă] APAC [;com]
Operanzi : nici unul
Cuvinte : 1
Execuţie: (PC)+1→PC
(ACC) + (registrul P deplasat) →ACC
Influenţează OV ; este influenţat de PM şi OVM.
Influienţează C.
Nu este influenţat de SXM.
Descriere: Conţinutul registrului P este deplasat după cum defines biţii de stare PM şi adunat la conţinutul acumulatorului. Rezultatul este depus în accumulator. APAC nu este influienţata de bitul SXM al registrului de stare; registrul P are întotdeauna semn extins.

Instructiunea APAC este subset al instrucţiunulor LTA, LTD, MAC, MACD, MPZA şi SQRA.

Exemplu: APAC ;(PM=0)

		Înaintea execuţiei			După execuţie
P		2629h	P		2629h
ACC	X	310h	ACC	0	2939h
	C			C	

B- salt necondiţionat

Sintaxa : [etichetă] B pma[,(ind)[,ARD următor]] [;com]
Operanzi : 0 ≤ pma ≤ 65535
0 ≤ ARP următor ≤ 7
Cuvinte : 2
Execuţie: pma→PC

Modifică AR(ARP) şi ARP conform specificaţiei.

Descriere: Registrul auxiliar curent şi ARP sunt modificate conform specificaţiei, iar controlul este transferat la adresa de memorie de program desemnată (pma). Este de remarcat că nu apar modificări ale AR sau ARP dacă nu se specifică nimic în acele cămpuri pma poate fi sau o adresă simbolică, sau una numerică.

Exemplu: B 0E81h ;număratorul de program este încărcat cu 0E81h, ceea ce determină continuarea programului de la acea locaţie

BLKD – mută bloc din memoria de date în memoria de date

Sintaxa : ad.dir.: [etichetă] BLKD dma1,dma2 [;com]
ad.dir.: [etichetă] BLKD dma1, {ind}[,ARD următor] [;com]

Operanzi : $0 \leq dma1 \leq 65535$
$0 \leq dma2 \leq 127$
$0 \leq$ ARP următor ≤ 7

Cuvinte : 2

Execuţie: (PC)+2→PC
(PFC)→MCS
Dma1→PFC
Dacă (controlul de repetare) ≠0, atunci
(dma1, adresată de PFC)→dma2,
Modifica AR(ARP) şi ARP conform specificaţiei,
(PFC)+1→PFC,
(controlul de repetare)-1→controlul de repetare.

În caz contrar:
(dma1, adresată de PFC)→dma2,
Modifică AR(ARP) şi ARP conform specificaţiei.

Descriere: Cuvinte consecutive de memorie sunt mutate dintr-un bloc sursă de memorie de date intr-un bloc de destinaţie de memorie de date. Adresa initial (cea mai mică) a blocului sursă este definite de al doilea cuvât al instrucţiunii. Adresa initială a blocului destinaţie este definite fie de dma2 conţinută în codul instrucţiunii (la adresa directă), fie de registrul AR curent (la adresa indirectă). În cazul modului de adresare indirectă, atât AR cât şi ARP pot fi modificaţi ca de obicei. În cazul modului de adresare direct dma2 este utilizată ca adresă destinaţie pentru mutarea blocului, dar nu este modificată la execuţia repetată a instrucţiunii. De aceea conţinutul memorie la adresa dma2 va fi acelaşi cu conţinutul memoriei la ultima adresă dma1 din secvenţa repetată.

Dacă trebuie mutate mai multe cuvinte, este necesară folosirea instrucţiunilor RPT sau RPTK împreună cu BLKD în modul de adresare indirectă. Numărul de cuvinte ce vor fi mutate este cu unul mai mare decăt numărul conţinut în controlul de repetări, RPTC, la începutul instrucţiunii. La sfârsitul acestei instrucţiuni, RPTC conţine zero şi , dacă se foloseşte adresarea indirectă, AR(ARP) va fi modificat şi va conţine adresa de după sfârşitul blocului destinaţie.

Este de remarcat că nu este necesar ca blocurile sursă şi destinaţie să fie în întregime pe cip sau externe. Totuşi BLKD nu poate fi folosită pentru a transfera date dintr-un registru mapat în memorie în locaţii ale memoriei de date.

După execuţie, PC pointează la instrucţiunea următoare lui BLKD.

Pe durata execuţiei unei operaţii BLKD cu RPT sau RPTK, întreruperile sunt inhibate.

Exemplu: RPTK 2
BLKD 30h,*+ ;dacă registrul auxiliar curent conţine valoarea 7.

Dma1

	Înaintea execuţiei		După execuţie
Memoria de date 30h	0h	Memoria de date 30h	0h
Memoria de date 31h	1h	Memoria de date 31h	1h
Memoria de date 32h	2h	Memoria de date 32h	2h

Dma2

	Înaintea execuţiei		După execuţie
Memoria de date 7h	567h	Memoria de date 7h	0h
Memoria de date 8h	94Fh	Memoria de date 8h	1h
Memoria de date 9h	6ED2h	Memoria de date 9h	2h

BLKP – mută bloc din memoria de program în memoria de date

Sintaxa : ad.dir.: [etichetă] BLKP pma,dma [;com]
ad.dir.: [etichetă] BLKP pma, {ind}[,ARP următor] [;com]
Operanzi : $0 \leq pma \leq 65535$
$0 \leq dma \leq 127$
$0 \leq$ ARP următor ≤ 7
Cuvinte : 2
Execuţie: (PC)+2→PC
(PFC)→MCS
pma→PFC
Dacă (controlul de repetare) ≠0, atunci
(pma, adresată de PFC)→dma,
Modifica AR(ARP) şi ARP conform specificaţiei,
(PFC)+1→PFC,
(controlul de repetare)-1→controlul de repetare.

În caz contrar:
(pma, adresată de PFC)→dma,
Modifică AR(ARP) şi ARP conform specificaţiei.

Descriere: Cuvinte consecutive de memorie sunt mutate dintr-un bloc sursă de memorie de program intr-un bloc de destinaţie de memorie de date. Adresa initial (cea mai mică) a blocului sursă este definite de al doilea cuvâ al instrucţiunii. Adresa initială a blocului destinaţie este definite fie de dma conţinută în codul instrucţiunii (la adresa directă), fie de registrul AR curent (la adresa indirectă). În cazul modului de adresare indirectă, atât AR cât şi ARP pot fi modificaţi ca de obicei. În cazul modului de adresare direct dma este utilizată ca adresă destinaţie pentru mutarea blocului, dar nu este modificată la execuţia repetată a instrucţiunii. De aceea conţinutul memorie la adresa dma va fi acelaşi cu conţinutul memoriei la ultima adresă pma din secvenţa repetată.

Dacă trebuie mutate mai multe cuvinte, este necesară folosirea instrucţiunilor RPT sau RPTK împreună cu BLKP în modul de adresare indirectă. Numărul de cuvinte ce vor fi mutate este cu unul mai mare decât numărul conţinut în controlul de repetări, RPTC, la începutul instrucţiunii. La sfârsitul acestei instrucţiuni, RPTC conţine zero şi , dacă se foloseşte adresarea indirectă, AR(ARP) va fi modificat şi va conţine adresa de după sfârşitul blocului destinaţie.

Este de remarcat că nu este necesar ca blocurile sursă şi destinaţie să fie în întregime pe cip sau externe. După execuţie, PC pointează la instrucţiunea următoare lui BLKP.
Pe durata execuţiei unei operaţii BLKP cu RPT sau RPTK, întreruperile sunt inhibate.

Exemplu: RPTK 2
BLKP 30h,*+ ;dacă registrul auxiliar curent conţine valoarea 70h.

pma

	Înaintea execuţiei		După execuţie
Memoria de date 30h	0h	Memoria de date 30h	0h
Memoria de date 31h	1h	Memoria de date 31h	1h
Memoria de date 32h	2h	Memoria de date 32h	2h

Dma

	Înaintea execuţiei		După execuţie
Memoria de date 70h	567h	Memoria de date 70h	0h
Memoria de date 71h	94Fh	Memoria de date 71h	1h
Memoria de date 72h	6ED2h	Memoria de date 72h	2h

BV – salt dacă există depăşire

Sintaxa : [etichetă] BV pma[, {ind}[,ARP următor]] [;com]
Operanzi : $0 \leq pma \leq 65535$
$0 \leq$ ARP următor ≤ 7
Cuvinte : 2
Execuţie: Dacă bitul de depăşire OV=1
Atunci pma→PC şi 0→OV
În caz contrar, (PC)+2→PC
Modifica AR(ARP) şi ARP conform specificaţiei,
Influenţează OV; influenţată de OV.
Descriere: Registrul auxiliar current şi ARP sunt modificate conform specificaţiei, iar semaforul de depăşire este curăţat. Dacă semaforul de depăşire OV=1, controlul este transferat la adresa de memorie de program desemnată (pma). În caz contrar se transferă controlul instrucţiunii următoare. Nu apar modificări ale AR sau ARP dacă nu se specifică nimic în acel câmp. Pma poate fi sau o adresă numerică, sau una simbolică.

Exemplu: BV OF50h ;dacă OV=1, se încarcă OF50h în numărătorul de program şi OV
;este curăţat; în caz contrar, numărătorul de program este
;incrementat cu 2.

CMPL – complementează acumulatorul

Sintaxa : [etichetă] CMPL [;com]
Operanzi : nici unul
Cuvinte : 1

Execuţie: (PC)+1→PC
(AC)→ACC;
Descriere: Conţinutul acumulatorului este înlocuit cu inversul său logic (complementul său faţă de 1)
Exemplu: CMPL

		Înaintea execuţiei			După excuţie	
ACC	X	1F518854h	ACC	X	0E0A877ABh	
	C			C		

CNFD – configurează blocul B0 ca memorie de date

Sintaxa : [etichetă] CNFD [;com]
Operanzi : nici unul
Cuvinte : 1
Execuţie: (PC)+1→PC
0→CNF(bitul de control al configuraţiei memoriei RAM de pe cip) influenţează CNF.
Descriere: Blocul 0 de memorie RAM de pe cip este configurat ca memorie de date. Blocul este mapat În memoria de date, între adresele 512 şi 767. CNFD este complementară instrucţiunii CNFP şi setează pe 0 bitul CNF din registrul de stare ST1. CNF mai poate fi încărcat folosind instrucţiunule CNFP şi LST1.
La TMS320C25, următoarele două instrucţiuni de după CNFD sau CNFP folosesc vechea valoare CNF.
La TMS320C26, această instrucţiune nu este validă si este nedefinită

Exemplu: CNFD ; se încarcă un zero în bitul de stare CNF şi astfel se configurează ;blocul B0 ca memorie de date

CNFP– configurează blocul B0 ca memorie de program

Sintaxa : [etichetă] CNFP [;com]
Operanzi : nici unul
Cuvinte : 1
Execuţie: (PC)+1→PC
0→CNF(bitul de control al configuraţiei memoriei RAM de pe cip) influenţează CNF.
Descriere: Blocul 0 de memorie RAM de pe cip este configurat ca memorie de program.
Blocul este mapat în memoria de program, între adresele 65280 şi 65535. CNFP este complementară instrucţiunii CNFD şi setează pe 1 bitul CNF din registrul de stare ST1. CNF mai poate fi încărcat folosind instrucţiunule CNFD şi LST1.
Configurarea acestui bloc ca memorie de programpermite folosirea numărătorului de program ca generator de adrese pentru accesarea datelor din memoria RAM de pe cip. Folosirea împreuna cu instrucţiunile de repetare, aceasta permite ca două locaţii de memories ă fie adresate simultan, una prin registrul auxiliar şi alta prin număratorul de program. Instrucţiunile care pot beneficia de acest avantaj sunt MAC,MACD BLKD şi BLKP.
La TMS320C25, următoarele două instrucţiuni de după CNFP sau CNDP folosesc vechea

valoare CNF.
La TMS320C26, această instrucţiune nu este validă si este nedefinită

Exemplu: CNFD ; se încarcă un unu în bitul de stare CNF şi astfel se configurează ;blocul B0 ca memorie de program.

EINT– permite întreruperile

Sintaxa : [etichetă] EINT [;com]
Operanzi : nici unul
Cuvinte : 1
Execuţie: (PC)+1→PC
0→ INTM(bitul de stare a modului de întrerupere).
Influenţează INTM
Descriere: Bitul de stare a modului de întrerupere (INTM) este setat pe 0 logic. Întreruperile mascabile sunt permise imediat după executarea instructiunii care urmează după EINT. Aceasta permite unei rutine de servire a unei întreruperi să permita întreruperile şi să execute instrucţiunea RET, înainte de a prelucra orice altă întrerupere activă în acel moment. Este de remarcat că instructiunea LST nu influienţează INTM.

Exemplu: EINT ;întreruperile mascabile sunt permise, iar INTM este ;setat pe zero.

IN– introduce date de la port

Sintaxa : ad.dir.: [etichetă] IN dma,PA [;com]
ad.ind.: [etichetă] IN {ind},PA[,ARP următor] [;com]
Operanzi : $0 \leq dma \leq 127$
$0 \leq$ ARP următor ≤ 7
$0 \leq$ adresă de port (PA) ≤ 15
Cuvinte : 1
Execuţie: (PC)+1→PC
Adresă de port → magistrala de adrese A3-A4
Magistrala de date D15-D0 → dma

Descriere: Instrucţiunea IN citeşte o valoare de 16 biti de la unul din din porturile externe de intrare/ iesire şi o depunere în locaţia de memorie de date specificată. Linia IS trece în logic pentru a indica o operaţie de intrare/iesire, iar formele semnalelor pe liniile STRB, R/W şi READY sunt identice cu cele de la o citire din memoria externă de date.

Exemplu: IN 5,PA3 ;(DP=5)
Sau
IN *,PA3 ;dacă registrul auxiliar curent conţine 645;
;citeşte un cuvânt de la perifericul plasat la adresa de
;port 3 şi îl depune în memoria de date, la adresa 645.

LAC – încearcă acumulatorul, cu deplasarea

Sintaxa : ad.dir.: [etichetă] LAC dma[,deplasare] [;com]
ad.ind.: [etichetă] LAC (ind){,deplasare [,ARP următor]] [;com]
Operanzi : $0 \leq dma \leq 127$

0 ≤ ARP următor ≤ 7
0 ≤ deplasare ≤ 15(implicit=o)

Cuvinte : 1

Execuţie: (PC)+1→PC
(dma)x $2^{deplasare}$ → ACC
Dacă SXM=1 atunci (dma) va fi cu semn extins.
Dacă SXM=0 atunci (dma) nu va fi cu semn extins.
Este influenţată de SXM.

Descriere: Conţinutul locaţiei de memorie de date adresate este deplasat la stânga şi la dreapta în Acumulator. În timpul deplasării, biţii de ordin inferior se completează cu zero. Biţii de ordin superior sunt completaţi cu valoarea bitului dacă SXM=1 sau cu zero dacă SXM=0.

Exemplu:
```
LAC     5,3          ;(DP=5)
   Sau
LAC     *,3          ;dacă registrul auxiliar curent conţine 645
```

Memoria de date 645	Înaintea execuţiei		Memoria de date 645	După execuţie	
	100h			100h	
ACC	X	2h	ACC	0	800h
	C			C	

LACK – încearcă acumulatorul, cu un octet, imediat

Sintaxa : [etichetă] LACK constantă [;com]

Operand : 0 ≤ constantă ≤ 255

Cuvinte : 1

Execuţie: (PC)+1→PC
Constantă pozitivă de 8 biţi→ACC
Nu este influenţată de SXM

Descriere: Constanta de 8 biţi, aliniată la dreapta, este încarcată în acumulator. Cei mai semnificativi 24 biţi ai acumulatorului sunt facuţi 0 (adică, extinderea semnului este suprimată).

Exemplu: LACK 0Dh

	Înaintea execuţiei			După excuţie	
ACC	X	2195183h	ACC	X	0Dh
	C			C	

LALK – încearcă acumulatorul, cu un cuvânt, imediat, cu deplasare

Sintaxa : [etichetă] LALK constantă[,deplasare] [;com]

Operanzi : constantă de 16 biţi
0 ≤ deplasare ≤15 (implicit =0)

Cuvinte : 2

Execuţie: (PC)+1→PC
Dacă SXM=1, atunci -32768 ≤ constanta ≤ 32767
Dacă SXM=0, atunci 0 ≤ constanta ≤ 65535
Este influenţată de SXM.

Descriere: Valoarea imediată de 16 biţi,deplasată la stânga conform specificaţiei, este încărcată în acumulator . Dacă SXM=1, constanta de 16 biţi deplasată va avea semnul extins; dacă SXM=0, biţi de ordin superior ai acumulatorului, rămaşi după deplasarea constantei, sunt Făcuţi zero. Este de remarcat că cel mai semnificativ bit al acumulatorului poate deveni 1

Numai dacă SXM=1 şi se încarcă un număr negativ. Numărătorul de deplasare este optional şi când lipseşte, are implicit valoarea zero.

Exemplu: LALK 0FE11h,4 ;(SXM=1)

		Înaintea execuţiei			După excuţie	
ACC	X	219Ch		ACC	X	0FFFFE110h
	C				C	

LALK 0FE11h,4 ;(SXM=0)

		Înaintea execuţiei			După excuţie	
ACC	X	219Ch		ACC	X	0FE110h
	C				C	

LAR – încearcă registrul auxiliar

Sintaxa : ad.dir.; [etichetă] LAR AR,dma [;com]
ad.ind.; [etichetă] LAR AR,{ind}[,ARP următor] [;com]

Operanzi : $0 \le dma \le 127$
$0 \le AR(registrul\ auxiliar) \le 7$
$0 \le ARP\ următor \le 7$

Cuvinte : 1

Execuţie: (PC)+1→PC
(dma)→registrul auxiliar AR

Descriere: Conţinutul adresei de memorie de date specificate este încărcat în registrul auxiliar desemnat : (AR).
Instrucţiunile LAR si SAR (memorează registrul auxiliar) pot fi folosite pentru încărcarea si memorarea registrelor auxiliare în timpul apelurilor de subrutine si a întreruperilor. Dacă un registru auxiliar nu este folosit pentru adresare indirectă, LAR şi SAR permit ca registrul să fie folosit ca şi registru suplimentar de memorare, mai ales pentru schimbarea valorilor între locaţii ale memoriei de date fără influenţarea conţinutului acumulatorului.
În modul de adresare indirectă, LAR nu realizează nici o modificare a registrului auxiliar curent marcat de ARP, dacă acesta este acelaşi cu registrul auxiliar specificat în instrucţiune.

Exemplu: LAR AR2,5 ;(DP=5, ARP=2)
Sau
LAR AR2,* ;dacă registrul auxiliar curent conţine 645

	Înaintea execuţiei		După execuţie
Memoria de date 645	100h	Memoria de date 645	100h
AR2	2h	AR2	800h

LARK – încarcă registrul auxiliar cu un octet, imediat

Sintaxa : [etichetă] LARK AR,constantă [;com]

Operanzi : $0 \leq$ constantă ≤ 255
$0 \leq$ AR(registrul auxiliar) ≤ 7
Cuvinte : 1
Execuţie: (PC)+1→PC
Constantă pozitivăde 8 biţi→ AR
Descriere: Valoarea imediată pe 8 biţi este încărcată ăn registrul auxiliar AR, aliniată la dreapta şi cu extensia de semn suprimată (adică octeţul superior este zero).
LARK este utilă pentru încărcarea valorii iniţiale a contorului de buclă într-un registru auxiliar, când se foloseşte instrucţiunea BANZ.

Exemplu: LARK AR3,OE5h

AR3	Inaintea execuţiei		După execuţie
	OF370h	AR3	OE5h

LARP – încarcă pointerul registrului auxiliar

Sintaxa : [etichetă] LARP constantă [;com]
Operanzi : $0 \leq$ constantă ≤ 7
Cuvinte : 1
Execuţie: (PC)+1→PC
(ARP) →ARB
Constantă → ARP
Influenţeză ARP şi ARB.
Descriere: Pointerul registrului auxiliar este încărcat cu conţinutul celor mai puţin semnificativi trei biţi ai instrucţiunii, care reprezintă o constantă ce identifică registrul auxiliar dorit. Vechiul ARP este copiat în câmpul ARB ai registrului de stare ST1. ARP mai poate fi modificat de instrucţiunile LST, LST1 şi MAR, precum şi de orice instrucţiune utilizată în modul de adresare indirect. Instrucţiunea LARP formează un subset al instrucţiunii MAR, codul ei fiind acelaşi cu al instrucţiunii MAR în modul de adresare indirectă. Instrucţiunea
MAR, *constantă
are acelaşi efect ca şi LARP.
Exemplu: LARP 3 ;orice instrucţiune următoare va folosi registrul auxiliar AR3
;pentru adresare indirectă.

LDP – încarcă pointerul paginii memoriei de date

Sintaxa : ad.dir.; [etichetă] LDP dma [;com]
ad.ind.; [etichetă] LDP {ind}[,ARP următor] [;com]
Operanzi : $0 \leq$ dma ≤ 127
$0 \leq$ ARP următor ≤ 7
Cuvinte : 1

Execuţie: (PC)+1→PC
Cei mai puţin semnificativi nouă biţi ai (dma)→DP
Influenţeză DP.
Descriere: Cei mai puţini semnificativi nouă biţi ai conţinutului locaţiei de memorie de date adresate sunt încărcaţi în registrul DP(ai pointerului la pagina memoriei de date). DP şi adresa memoriei de date pe 7biţi sunt concatenate pentru a forma adresa memoriei de date pe 16 biţi. DP mai poate fi incărcat cu instrucţiunile LST şi LDPK.
Exemplu: LDP 5 ;(DP=5)
Sau

LDP * ;dacă registrul auxiliar curent conţine 645.

	Înaintea execuţiei		După execuţie
Memoria de date 645	0EB25h	Memoria de date 645	0EB25h
DP	42h	DP	125h

LDPK – încarcă pointerul paginii memoriei de date, imediat

Sintaxa : [etichetă] LDPK constantă [;com]

Operanzi : 0 ≤ constantă≤ 511
Cuvinte : 1
Execuţie: (PC)+1→PC
constanta→DP
Influenţeză DP.
Descriere: Registrul DP (al pointerului la pagina memoriei de date) este încărcat cu o costantă de 9 biţi. DP şi adresa memoriei de date pe 7 biţi sunt concatenate pentru a forma adresa memoriei de date pe 16 biţi.
DP≥8 specifică memoria de date externă. DP cuprins între 4 şi 7 specifică blocurile B0 sau B1 de RAM de pe cip. Blocul B2 este plasat în ultimele 32 de cuvinte superioare ale paginii 0. DP mai poate fi încărcat cu instrucţiunile LST şi LDP.

Exemplu: LDPK 14h ;pointerul la pagina de date este încărcat cu 20 (14h).

LRLK – încarcă registrul auxiliar cu un cuvânt, imediat

Sintaxa : [etichetă] LRLK AR,constantă [;com]

Operanzi : 0 ≤ AR (registrul auxiliar)≤7
0 ≤ constantă≤ 65535
Cuvinte : 2
Execuţie: (PC)+1→PC
constanta→Ar
nu este influenţată de SXM; nu influenţează SXM.
Descriere: Valoarea imediată de 16 biţi este încărcată în registrul auxiliary specificatde câmpul AR. Constanta specificată trebuie să fie un întreg fără semn, iar valoarea ei nu este influienţată de SXM.
Exemplu: LRLK AR,2836h

	Inaintea execuţiei		După execuţie
AR7	OF3Ah	AR7	2836h

LST – încarcă registrul de stare ST0

Sintaxa : ad.dir.: [etichetă] LST dma [;com]
ad.ind.: [etichetă] LST {ind}[,ARP următor] [;com]

Operanzi : $0 \leq dma \leq 127$
$0 \leq$ ARP următor≤ 7
Cuvinte : 1
Execuţie: (PC)+1→PC
(dma)→registrul de stare STO
Influenţează ARP,OV,OVM şi DP.
Nu influenţează INTM sau ARB.
Descriere: Registrul de stare ST0 este încărcat cu valoareadin locaţia de memorie de date adresată. Bitul INTM (semaforul modului de întrerupere) nu este influenţat de LST. Nici ARB nu este influenţat chiar dacă se încarcă un nou ARP.Este ignorată în acelşi timp, orice valoare următoare a ARP specificată prin modul de adresare indirect. În schimb, ARP este încărcat cu valoarea conţinută în cuvântul de memorie de dare adresat. Instrucţiunea LST este utilizată pentru încărcarea registrului de stare ST0 după întreruperi sau apeluri de subroutine. ST0 conşine biţi de stare:OV(semafor de depăşire), OVM(bitul modului de depăşire), INTM(semaforul modului de întrerupere), ARP(pointerul la registrul auxiliar) şi DP(pointerul la pagina de memorie de date). Aceşti biţi au fost memoraţi(cu instrucţiunea SST) în cuvântul de memorie de date astfel:

15 14 13	12	11 10	9	8 7 6 5 4 3 2 1 0
ARP	OV	OVM	1	INTM DP

Exemplu:
```
LST     5       ;(DP=5)
        Sau
LST     *,1     ;dacă registrul auxiliar curent conţine 645;
                ;cuvântul de memorie de date adresat este încărcat în
                ;registrul de stare ST0, cu excepţia bitului INTM. Chiar
                ;dacă este specificată valoarea următoare a ARP, acea
                ;valoare este ignorată şi chiar dacă se încarca un nou
                ;ARP, vechiul ARP nu este încărcat în ARB.
```

	Înaintea execuţiei		După execuţie
Memoria de date 645	0EB700h	Memoria de date 645	0E700h
ST0	6462h	ST0	OE500h
ST1	3180h	ST1	3180h

LST 1– încarcă registrul de stare ST1

Sintaxa : ad.dir.: [etichetă] LST1 dma [;com]
ad.ind.: [etichetă] LST1 {ind}[,ARP următor] [;com]

Operanzi : $0 \leq dma \leq 127$
$0 \leq$ ARP următor≤ 7
Cuvinte : 1
Execuţie: (PC)+1→PC
(dma)→registrul de stare ST1
(ARB)→ARP
Influenţează ARP, ARB,CNF, TC, SXM, XF, FO, TXM şi PM.

Influenţează C, HM, şi FSM (TMS320C25).

Descriere: Registrul de stare ST1 este încărcat cu valoarea din locaţia de memorie de date adresată.Biţii valorii care se încarcă în ARB sunt încărcţi , de asemenea, şi în ARP, pentru a simplifica comutările de context. Este ignorată, în acelaşi timp orice valoare următoare a ARP specificată prin modul de adresare indirect.

Intrucţiunea LST1 este utilizată pentru încărcarea biţilor de stare după întreruperi sau apeluri de subroutine. ST1 conţine următorii biţi de stare : ARB(bufferul pointerului la registrul auxiliar), CNF (bitul de control al configuraţiei memoriei RAM de pe cip), TC(semaforul test/control), SXM(modul de extensie al semnului), XF(bitul de stare a pinului XF), FO(formatul portului serial), TXM(mod de transmisie) şi PM(modul de deplasare a registrului produs). Pe TMS320C25, ST1 mai conţine şi biţii de stare : C(transport(carry)), HM(mod hold) şi FSM(mod de sincronizare a cadrului). Biţii încărcaţi in ST1 din cuvântul adresat al memoriei de date sunt aranjaţi ca mai jos:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARB			CNF	TC	SXM	C	1	1	HM	FSM	XF	FO	TXM	PM	

Exemplu:
```
LST1    5       ;(DP=5)
        Sau
LST1    *,1     ;dacă registrul auxiliar curent conţine 645;
                ;cuvântul de memorie de date adresat este încărcat în
                ;registrul de stare ST1
```

	Înaintea execuţiei		După execuţie
Memoria de date 645	0E7A3h	Memoria de date 645	0E7A3h
ST1	3180h	ST1	OE7A3h
ST0	462h	ST0	OE462h

LT – încarcă registrul T

Sintaxa :
```
ad.dir.:    [etichetă]  LT  dma                 [;com]
ad.ind.:    [etichetă]  LT  {ind}[,ARP următor] [;com]
```

Operanzi : $0 \le dma \le 127$
$0 \le$ ARP următor≤ 7

Cuvinte : 1

Execuţie: (PC)+1→PC
(dma)→registrul T

Descriere: Registrul T este încărcat cu conţinutul locaţiei de memorie de date specificate. Instructiunea LT poate fi folosită în pregătirea inmulţirii, la încărcarea registrului T. Vezi instrucţiunile LTA, LTD, LTP, LTS, MPY, MPYK, MPZA, MPYS ŞI MPYU.

Exemplu:
```
LT      5       ;(DP=5)
        Sau
LT      *       ;dacă registrul auxiliar curent conţine 645;
```

	Înaintea execuţiei		După execuţie
Memoria de date 645	100h	Memoria de date 645	100h
T	3E6Ah	T	100h

LTA – încarcă registrul T şi acumulează produsul precedent

Sintaxa : ad.dir.: [etichetă] LTA dma [;com]
ad.ind.: [etichetă] LTA {ind}[,ARP următor] [;com]

Operanzi : $0 \leq dma \leq 127$
$0 \leq$ ARP următor≤ 7
Cuvinte : 1
Execuţie: (PC)+1→PC
(dma)→registrul T
(ACC)+(registrul P deplasat)→ACC
Influenţează OV; este influenţată de OVM şi PM.
Influenţează C
Descriere: Registrul T este încărcat cu conţinutul locaţiei de memorie de date specificate Conţinutul registrului produs este deplasat conform programării biţilor PM şi adunat la acumulator. Rezultatul adunării este depus în accumulator.
Funcţia instructiunii LTA este inclusă în instrucţiunea LTD.

Exemplu: LT 5 ;(DP=5, PM=0)
Sau
LT * ;dacă registrul auxiliar curent conţine 645;

	Înaintea execuţiei		După execuţie
Memoria de date 645	100h	Memoria de date 645	100h
T	3E6Ah	T	100h
P	1000h	P	1000h
ACC	X 2h	ACC	0 1002h
	C		C

LTD – încarcă registrul T , acumulează produsul precedent şi mută data

Sintaxa : ad.dir.: [etichetă] LTD dma [;com]
ad.ind.: [etichetă] LTD {ind}[,ARP următor] [;com]

Operanzi : $0 \leq dma \leq 127$

0 ≤ ARP următor≤ 7

Cuvinte : 1

Execuţie: (PC)+1→PC
(dma)→registrul T
(dma)→dma+1
(ACC)+(registrul P deplasat)→ACC
Influenţează OV; este influenţată de OVM şi PM.
Influenţează C

Descriere: Registrul T este încărcat cu conţinutul locaţiei de memorie de date specificate. Conţinutul registrului produs este deplasat conform programării biţilor PM şi adunat la acumulator. Rezultatul adunării este depus în acumulator. Conţinutul locaţiei de memorie de date specificate este copiat, de asemenea, şi în conţinutul adresei superioare următoare.
Această instrucţiune este validă permanent numai pentru blocurile B1 şi B2; Este validă şi pentru blocul B0, dacă acesta este configurat ca şi memoriede date.
În acest caz, funcţia de mutare a datelor este continua peste marginile blocurilor B0 şi B1, adică lucrează la fel pentru toate locaţiile cuprinse între adresele 512 şi 1023. Funcţia de mutare a datelor nu poate fi folosită cu memoria externă de date sau cu registrele mapate în memorie. Aceasta instructiune este descrisă la instructiunea DMOV. Este de remarcat că funcţia instrucţiunii LTD utilizate cu memoria externă de date este identică cu cea a instrucţiunii LTA.

Exemplu:
```
LT      5       ;(DP=5, PM=0)
        Sau
LT      *       ;dacă registrul auxiliar curent conţine 645;
```

	Înaintea execuţiei		După execuţie
Memoria de date 645	100h	Memoria de date 645	100h
Memoria de date 646	OE49Fh	Memoria de date 646	100h
T	3E6Ah	T	1000h
P	1000h	P	1000h
ACC	X: 2h	ACC	0: 1002h
	C		C

LTP – încarcă registrul T şi memorează registrul P în acumulator

Sintaxa : ad.dir.: [etichetă] LTP dma [;com]
ad.ind.: [etichetă] LTP {ind}[,ARP următor] [;com]

Operanzi : 0 ≤ dma≤127
0 ≤ ARP următor≤ 7

Cuvinte : 1

Execuţie: (PC)+1→PC
(dma)→registrul T
(registrul P deplasat)→ACC
Este influenţată de PM.

Descriere: Registrul T este încărcat cu conţinutul locaţiei de memorie de date specificate. Conţinutul registrului produs este deplasat conform programării biţilor PM şi depus în acumulator.

Exemplu: LTP 5 ;(DP=5, PM=0)

Sau

LTP * ;dacă registrul auxiliar curent conţine 645;

	Înaintea execuţiei		După execuţie
Memoria de date 645	100h	Memoria de date 645	100h
T	3E6Ah	T	100h
P	1000h	P	1000h
ACC	X: 2h; C	ACC	X: 1000h; C

LTS – încarcă registrul T şi memoscade produsul precedent

Sintaxa : ad.dir.: [etichetă] LTS dma [;com]

ad.ind.: [etichetă] LTS {ind}[,ARP următor] [;com]

Operanzi : $0 \leq dma \leq 127$

$0 \leq$ ARP următor≤ 7

Cuvinte : 1

Execuţie: (PC)+1→PC

(dma)→registrul T

(ACC)-(registrul P deplasat)→ACC

Influenţează OV; este influenţată de OVM şi PM.

Influenţează C

Descriere: Registrul T este încărcat cu conţinutul locaţiei de memorie de date specificate. Conţinutul registrului produs este deplasat conform programării biţilor PM şi Scăzut din acumulator. Rezultatul scăderii este depus în accumulator.

Exemplu: LTS 5 ;(DP=5, PM=0)

Sau

LTS * ;dacă registrul auxiliar curent conţine 645;

	Înaintea execuţiei		După execuţie
Memoria de date 645	100h	Memoria de date 645	100h
T	3E6Ah	T	100h
P	1000h	P	1000h
ACC	X: 2h; C	ACC	0: OFFFFF002h; C

MAC **înmulţeşte şi acumulează**

sintaxă: ad.dir.: [etichetă] MAC pma,dma [;com]
ad.ind.: [etichetă] MAC pma,{ind}[,ARP următor] [;com]

operanzi: $0 \leq$ pma ≤ 65535
$0 \leq$ dma ≤ 127
$0 \leq$ ARP următor ≤ 7

cuvinte: 2

execuţie: TMS320C25
(PC)+2 → PC
(PFC) → MCS
(pma) → PFC
Dacă (contor de repetare) ≠ 0 , atunci
(ACC) +(registrul P deplasat) → ACC ,
(dma) → registrul T,
(dma) x (pma,adresat de PFC) → registrul P,
Modifică AR(ARP) şi ARP conform specificaţiei ,
(PFC)+1 → PFC,
(contor de repetare) -1 → contor de repetare .
În caz contrar ,
(ACC) +(registrul P deplasat) → ACC ,
(dma) → registrul T,
(dma) x (pma,adresat de PFC) → registrul P,
Modifică AR(ARP) şi ARP conform specificaţiei ,
(MCS) →PFC.
Influenţează C şi OV; influenţată de OVM şi PM .

descriere : Instrucţiunea MAC înmulţeşte o valoare din memoria de date (specificată de dma) cu o valoare din memoria de program (specificată de pma).Adună , de asemenea , la acumulator produsul anterior deplasat conform programării biţilor PM.
Locaţiile din memoriile de date şi de program pot fi , la TMS320C25 , orice locaţii nerezervate ale memoriei interne sau externe .Dacă memoria de program este blocul B0 al memoriei RAM interne , atunci bitul CNF trebuie să fie 1, iar cei mai semnificativi 8 biţi ai pma trebuie să fie 0FFh.
În cazul modului de adresare directă , adresa locaţiei din memoria de date nu poate fi modificată în timpul unei execuţii repetate a instrucţiunii .
Când instrucţiunea MAC este executată repetat , adresa memoriei de program , conţinută în PFC , este incrementată cu 1 în timpul fiecărui ciclu executat . Acest fapt permite accesarea simplă a unei serii de operanzi plasaţi în memorie .
Instrucţiunea MAC este utilă pentru calcularea sumelor lungi de produse.

exemplu:
```
RPTK  2          ;realizează o suma de 2 produse dintre o constantă
                 ;plasată în memoria de date la adresa 645 şi valori
                 ;plasate în memoria de program începând de la adresa 0FF00h
MAC  0FF00h,5    ;(DP=5 ,PM=0,CNF=1,ACC=0,P=0)
```

*operanzii implicaţi sunt:

	înaintea execuţiei	după execuţie
memoria de date 645	2h	2h
memoria de program 0FF00h	10h	2h
memoria de program 0FF001h	20h	2h
memoria de program 0FF02h	30h	2h

*după prima execuţie se obţine primul produs , în P;

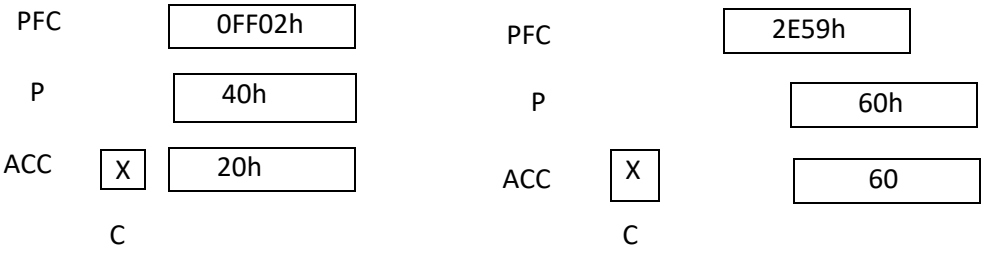
RPTC		2h	RPTC	1h
PFC		2E59h	PFC	0FF01h
P		0h	P	20h
ACC	X	0h	ACC	X 0h
	C			C

*după a doua execuţie se obţin al doliea produs , în P , şi primul termen al sumei, în acumulator

RPTC		1h	RPTC	0h
PFC		0FF01h	PFC	0FF02h
P		20h	P	40h
ACC	X	0h	ACC	0 20h
	C			C

*după a treia execuţie se obţine suma dorită , în acumulator ,se obţine, redundant , şi al treilea produs

RPTC		0h	RPTC	0h



MACD **înmulţeşte şi acumulează, cu mutarea datei**

sintaxă: ad.dir.: [etichetă] MACD pma,dma [;com]
ad.ind.: [etichetă] MACD pma,{ind}[,ARP următor] [;com]

operanzi: $0 \leq$ pma ≤ 65535
$0 \leq$ dma ≤ 127
$0 \leq$ ARP următor ≤ 7

cuvinte: 2

execuţie: TMS320C25
(PC)+2 → PC
(PFC) → MCS
(pma) → PFC
Dacă (contor de repetare) ≠ 0 , atunci
(ACC) +(registrul P deplasat) → ACC ,
(dma) → registrul T,
(dma) x (pma,adresat de PFC) → registrul P,
Modifică AR(ARP) şi ARP conform specificaţiei ,
(PFC)+1 → PFC,
(contor de repetare) -1 → contor de repetare .
În caz contrar ,
(ACC) +(registrul P deplasat) → ACC ,
(dma) → registrul T,
(dma) x (pma,adresat de PFC) → registrul P,
Modifică AR(ARP) şi ARP conform specificaţiei ,
(MCS) →PFC.
Influenţează C şi OV; influenţată de OVM şi PM .

descriere : Instrucţiunea MAC înmulţeste o valoare din memoria de date (specificată de dma) cu o valoare din memoria de program (specificată de pma).Adună , de asemenea , la acumulator produsul anterior deplasat conform programării biţilor PM.
Locaţiile din memoriile de date şi de program pot fi , la TMS320C25 , orice locaţii nerezervate ale memoriei interne sau externe .Dacă memoria de program este blocul B0 al memoriei RAM interne , atunci bitul CNF trebuie să fie 1, iar cei mai semnificativi 8 biţi ai pma trebuie să fie 0FFh.
În cazul modului de adresare directă , adresa locaţiei din memoria de date nu poate fi modificată în timpul unei execuţii repetate a instrucţiunii .
MACD functionează în acelaşi mod ca şi MAC, cu adăugarea suplimentară a facilitaţii de mutare a datelor pentru blocurile B0,B1 şi B2 (vezi instrucţiunea DMOV) .Din această cauză , dacă MACD adresează memoria externă sau unul din registrele mapate în memorie , efectul său va fi acelaşi cu al instrucţiunii MAC .

Când instrucţiunea MACD este executată repetat , adresa memoriei de program conţinută în PFC , este incrementată cu 1 în timpul fiecărui ciclu executat .

exemplu:

```
RPTK  2          ;realizează o suma de 2 produse dintre doua valori plasate
                 ;descendent în memoria de dateîincepând de la adresa 645
                 ;şi 2 valori plasate în memoria de program începând de la
                 ;adresa 0FF00h ; datele din memoria de date sunt plasate
                 ;ca şi la DMOV ;
                 ;(PM=0, CNF=1, ACC=0, P=0, ARP=3)
MACD  0FF00h,*-  ;dacă AR3=645
```

*după prima execuţie se obţine primul produs , în P , iar conţinutul locaţiei de date 645 este mutat în locaţia de date 646;

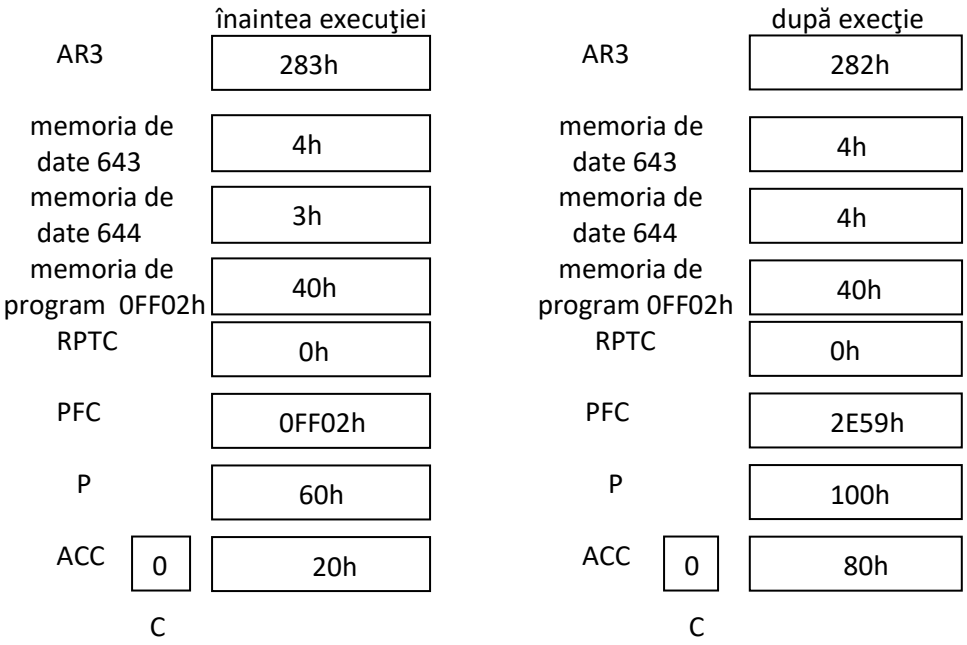
	înaintea execuţiei	după execţie
AR3	285h	284h
memoria de date 645	2h	2h
memoria de date 646	3619h	2h
memoria de program 0FF00h	10h	10h
RPTC	2h	1h
PFC	2E59h	0FF01h
P	0h	20h
ACC	0h	0h
C	X	0

*după a doua execuţie se obţin al doilea produs , în P, şi primul termen al sumei , În acumulator, conţinutul de date 644 este mutat în locaţia de date 645;

	înaintea execuţiei	după execţie
AR3	284h	283h
memoria de date 644	3h	3h
memoria de date 645	2h	3h
memoria de program 0FF01h	20h	20h
RPTC	1h	0h
PFC	0FF01h	0FF02h
P	20h	60h
ACC	0h	0h
C	0	0

*după a treia execuţie se obţine suma dorită în acumulator, se obţine, redundant,

şi al treilea produs; conţinutul locaţiei de date 643 este mutat în locaţia de date 644;



MAR	modifică registrul auxiliar

sintaxă: ad.dir.: [etichetă] MAR dma [;com]
ad.ind.: [etichetă] MAR {ind}[,ARP următor] [;com]

operanzi: $0 \leq$ dma ≤ 127
$0 \leq$ ARP următor ≤ 7

cuvinte: 1

execuţie: (PC)+1 → PC
Modifică ARP şi AR(ARP) conform specificaţiei din câmpul de adresare indirectă (în modul de adresare directă acţionează ca şi o instrucţiune NOP).

descriere: Instrucţiunea MAR acţionează , în modul de adresare direct, ca şi o instrucţiune în decursul căreia nu se efectueaza nici o operaţie.
În modul de adresare indirect se pot modifica registrul auxiliar curent şi ARP , însă nici în acest caz nu se foloseşte locaţia de memorie referită în instrucţiune.
MAR este utilizată numai pentru modificarea registrelor auxiliare sau a lui ARP .
Dacă se specifică o noua valoare pentru ARP , vechea lui valoare se copiază în câmpul ARB al registrului de stare ST1.Este de remarcat că orice operaţie executată de MAR se poate executa cu orice instrucţiune cu adresare indirectă.
Se poate remarca faptul că NOP şi LARP sunt forme particulare ale instrucţiunii MAR.

Exemplu:
```
MAR   5                ;este echivalentă cu un NOP
  Sau
MAR   *-,2             ;decrementează registrul auxiliar curent şi face ARP=2.
```

MPY	înmulţeşte

sintaxă: ad.dir.: [etichetă] MPY dma [;com]
ad.ind.: [etichetă] MPY {ind}[,ARP următor] [;com]

operanzi: $0 \leq$ dma ≤ 127
$0 \leq$ ARP următor ≤ 7

cuvinte: 1

execuţie: (PC)+1 → PC
(registrul T) x (dma) → registrul P

descriere: Conţinutul registrului T se înmulţeste cu conţinutul locaţiei de memorie de date adresate şi rezultatul este plasat în registrul P.

exemplu:
```
MPY  5                ;(DP=5)
  sau
MPY  *                ; dacă registrul auxiliar curent conţine 645
```

	înaintea execuţiei	după execţie
memoria de date 645	100h	100h
T	6Ah	6Ah
P	0F295h	6A00h

MPYA	înmulţeste şi acumulează produsul precedent

sintaxă: ad.dir.: [etichetă] MPY A dma [;com]
ad.ind.: [etichetă] MPYA {ind}[,ARP următor] [;com]

operanzi: 0 ≤ dma ≤ 127
0 ≤ ARP urmator ≤ 7

cuvinte: 1

execuţie: (PC)+1 → PC
(ACC)+(registrul P deplasat) → ACC
(registrul T) x (dma) → registrul P
Influenţează C şi OV ; este influenţată de OVM şi PM.

descriere: Conţinutul registrului T se înmulţeste cu conţinutul locaţiei de memorie de date adresate şi rezultatul este plasat în registrul P.De asemenea, se adună la acumulator produsul anterior deplasat conform programarii biţilor PM.

exemplu:
```
MPYA  5               ;(DP=5)
  sau
MPYA  *               ; dacă registrul auxiliar curent conţine 645
```

		înaintea execuţiei		după execţie
memoria de date 645		100h		100h
T		6Ah		6Ah
P		0F295h		6A00h
ACC	X	24AEh	0	11743h
	C		C	

MPYK	înmulţeste imediat

sintază: [etichetă] MPYK constantă [;com]

operand: -4096 ≤ constantă ≤ 4095
$-2^{12} \leq$ constantă $\leq 2^{12}-1$

cuvinte: 1

execuţie: (PC)+1 → PC

(registrul T) x constantă → registrul P

Nu este influenţată de SXM.

exemplu: MPYK -2

	înaintea execuţiei		după execuţie
T	0Ah	T	0Ah
P	0F295h	P	0FFFFFFDCh

MPYS	înmulţeşte şi scade produsul precedent

sintaxă: ad.dir.: [etichetă] MPY S dma [;com]

ad.ind.: [etichetă] MPYS {ind}[,ARP următor] [;com]

operanzi: 0 ≤ dma ≤ 127

0 ≤ ARP următor ≤ 7

cuvinte: 1

execuţie: (PC)+1 → PC

(ACC) - (registrul P deplasat) → ACC

(registrul T) x (dma) → registrul P

Influenţeaza C şi OV ; este influenţată de OVM şi PM.

descriere: Conţinutul registrului T se înmulţeste cu conţinutul locaţiei de memorie de date adresate şi rezultatul este plasat in registrul P. De asemenea, se scade din acumulator produsul anterior deplasat conform programarii biţilor PM.

exemplu: MPYS 5 ;(DP=5 , PM=0)

sau

MPYS * ; dacă registrul auxiliar curent conţine 645

		înaintea execuţiei			după execţie
memoria de date 645		100h	memoria de date 645		100h
T		6Ah	T		6Ah
P		95h	P		6A00h
ACC	X	24AEh	ACC	0	2419h
	C			C	

NEG	neagă acumulatorul

sintaxă: [etichetă] NEG [;com]

operanzi: nici unul

cuvinte: 1

execuţie: (PC)+1 → PC

(ACC) x (-1) → ACC

Influenţează OV ; este influenţată de OVM .

Influenţează C.

descriere: Conţinutul acumulatorului este înlocuit cu complementul sau aritmetic (complementul faţa de doi) . Bitul OV se face 1 dacă se cere negarea valorii 80000000h. Dacă OVM=1, conţinutul acumulatorului este înlocuit cu 7FFFFFFFh.Dacă OVM =0 , rezultatul este 80000000h. Bitul C este resetat la 0 pentru toate valorile diferite de 0 ale

acumulatorului şi este setat la 1 dacă acumulatorul este 0 .

exemplu: NEG

OR	SAU cu acumulatorul

sintaxă: ad.dir.: [etichetă] OR dma [;com]
ad.ind.: [etichetă] OR {ind}[,ARP următor] [;com]

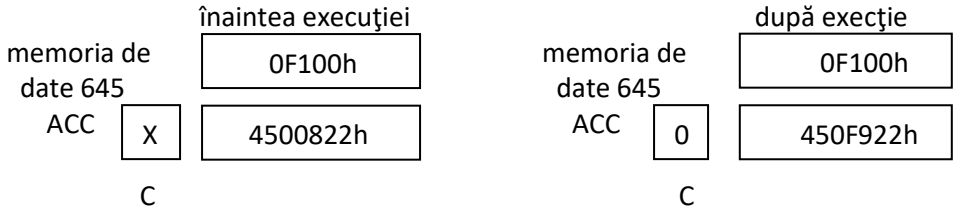
operanzi: $0 \leq$ dma ≤ 127
$0 \leq$ ARP următor ≤ 7

cuvinte: 1

execuţie: (PC)+1 → PC;
(ACC(15-0)) SAU (dma) → ACC(15-0);
ACC(31-16) → ACC(31-16).
Nu este influenţata de SXM.

descriere: Jumătatea inferioara a acumulatorului este supusă unei operaţii SAU cu conţinutul locaţiei de memorie de date adresate . Jumatatea superioară a acumulatorului este supusă unei operaţii SAU cu 0. De aceea , jumătatea superioară a acumulatorului este neschimbată după execuţia acestei instrucţiuni.

exemplu: OR 5 ;(DP=5)
sau
OR * ; dacă registrul auxiliar curent conţine 645



ORK	SAU imediat cu acumulatorul,cu deplasare

sintaxă: [etichetă] ORK constantă[,deplasare] [;com]

operanzi: constantă de 16 biţi
$0 \leq$ deplasare ≤ 15 (implicit = 0)

cuvinte: 2

execuţie: (PC)+2 → PC;
(ACC(30-0)) SAU [constantă x $2^{deplasare}$] → ACC(30-0);
ACC(31) → ACC(31).
Nu este influenţata de SXM.

descriere: Constanta imediată de 16 biţi este deplasată la stanga şi supusă unei operaţii SAU cu acumulatorul. Rezultatul este depus în acumulator . Biţii inferiori de sub şi biţii superiori de deasupra valorii deplasate sunt trataţi ca zerouri şi nu modifică , astfel , biţii corespunzători ai acumulatorului .Este de remarcat că cel mai semnificativ bit al acumulatorului nu este modificat , indiferent de valoarea deplasării .

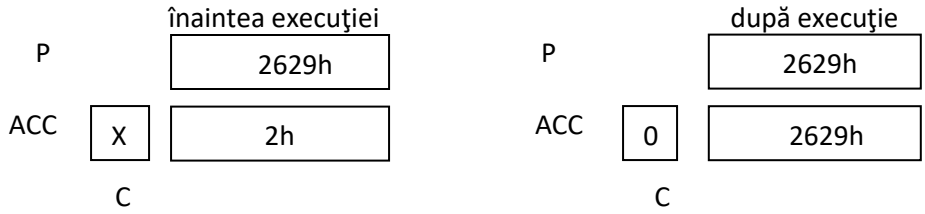
exemplu: ORK 0FFFFh,12

OUT	scoate date la port

sintaxă: ad.dir.: [etichetă] OUT dma,PA [;com]
ad.ind.: [etichetă] OUT {ind},PA,[,ARP următor] [;com]
operanzi: 0 ≤ dma ≤ 127
0 ≤ ARP următor ≤ 7
0 ≤ adresa port(PA) ≤ 15
cuvinte: 1
execuţie: (PC)+1 → PC;
adresa de port → magistrala de adrese A3-A0
0 → magistrala de adrese A15-A4
(dma) → magistrala de date D15-D0
descriere: Instrucţiunea OUT scrie o valoare de 16 biţi dintr-o locaţie de memorie de date la portul de intrare/ieşire specificat . Linia IS trece în 0 logic pentru a indica o operaţie de intrare/ieşire , iar formele semnalelor pe liniile STRB , R/W şi READY sunt identice cu cele de la o scriere în memoria externă de date.
exemplu:
```
OUT     5   ,PA3          ;(DP=5 )
  sau
OUT     *   ,PA3          ; dacă registrul auxiliar curent conţine 645;
                          ;scoate conţinutul adresei 645 a memoriei de
                          ;date la perifericul plasat la adresa de port 3
```

PAC	încarcă acumulatorul cu registrul P

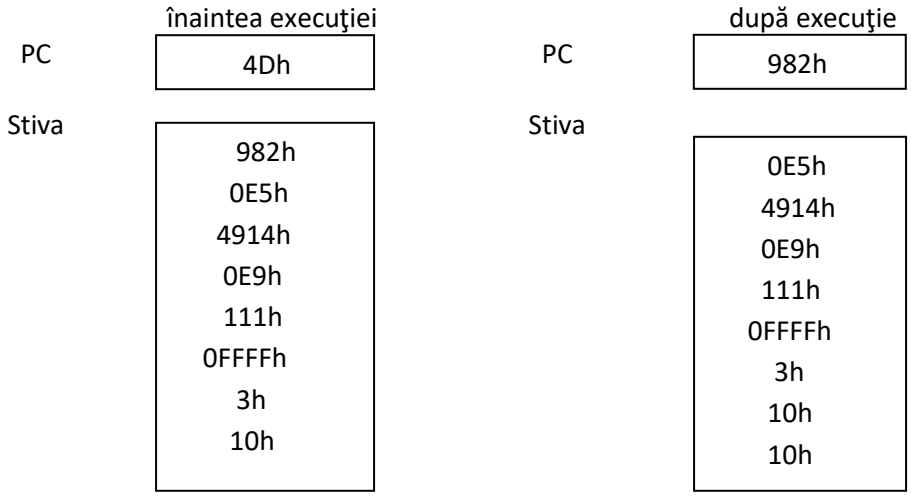
sintaxă: [etichetă] PAC [;com]
operanzi: nici unul
cuvinte: 1
execuţie: (PC)+1 → PC
(registrul P deplasat) → ACC
Influenţata de PM.
descriere: Conţinutul registrului P , deplasat conform programării biţilor PM , este încărcat în acumulator.
exemplu:
```
PAC             ;(PM=0)
```



RET	revenire din subrutină

sintaxă: [etichetă] RET [;com]
operanzi: nici unul
cuvinte: 1

execuţie: (TOS) → PC
Ridică stiva cu un nivel.
descriere: Conţinutul vârfului stivei (TOS) este copiat în contorul de program , iar stiva este apoi ridicată cu un nivel . RET este folosită împreună cu CALA si CALL pentru lucrul cu subutine .
exemplu: RET



ROL	roteşte acumulatorul la stanga

sintaxă: [etichetă] ROL [;com]
operanzi: nici unul
cuvinte: 1
execuţie: (PC)+1 → PC
ACC(31) → C
(ACC(30-0)) → ACC(31-1)
(C,de dinainte de ROL) → ACC(0)
Influenţează C .
Nu este influenţată de SXM.
descriere: Instrucţiunea ROL roteşte acumulatorul la stanga cu un bit . Bitul cel mai semnificativ este deplasat în bitul de transport (C), iar valoarea acestuia anterioară instrucţiunii ROL este deplasată în bitul cel mai puţin semnificativ .
exemplu: ROL

ROR	roteşte acumulatorul la dreapta

sintaxă: [etichetă] ROR [;com]
operanzi: nici unul
cuvinte: 1
execuţie: (PC)+1 → PC
(ACC(0)) → C
(ACC(31-0)) → ACC(30-1)
(C,de dinainte de ROR) → ACC(31)

Influenţează C .
Nu este influenţată de SXM.

descriere: Instrucţiunea ROR roteşte acumulatorul la dreapta cu un bit . Bitul cel mai puţin semnificativ este deplasat în bitul de transport (C), iar valoarea acestuia anterioară instrucţiunii ROR este deplasată în bitul cel mai semnificativ .

exemplu: ROR

ROVM	resetează modul depaşire

sintaxă: [etichetă] ROVM [;com]
operanzi: nici unul
cuvinte: 1
execuţie: (PC)+1 → PC
0 → bitul OVM din registrul de stare ST0
Influenţeaza OVM .

descriere: Bitul modului de depaşire(OVM) din registrul de stare ST0 este facut 0 , fapt ce produce inhibarea modului depaşire. Dacă apare o depaşire când OVM = 0, semaforul de depăşire (OV) se face 1 , iar rezultatul influenţat de depăşire se depune îîn acumulator .

exemplu: ROVM ;bitul OVM este făcut 0 logic , fapt ce determină inhibarea modului
;depăşire pentru toate operaţiile aritmetice următoare

RPTK	repetă instrucţiuni cât este specificat printr-o valoare imediată

sintaxă: [etichetă] RPTK constantă [;com]
operand: 0 ≤ constantă ≤ 255
cuvinte: 1
execuţie: (PC)+1 → PC
constantă → RPTC

descriere: Valoarea imediată de opt biţi este încarcată în contorul de repetare (RPTC) . Acest fapt face ca instrucţiunea urmatoare , dacă este repetabilă , să fie executată de un număr de ori mai mare cu unu decât valoare contorului de repetare . Întreruperile sunt mascate până la efectuarea numărului specificat de execuţii ale instrucţiunii următoare deoarece contorul de repetare nu poate fi salvat în cazul unei schimbări de context. Contorul de repetare este şters la iniţializarea hardware a procesorului (la reset). Instrucţiunile RPT şi RPTK sunt utile mai ales pentru repetarea instrucţiunilor ca BLKP , BLKD , IN , MAC , MACD , NORM , OUT , TBLR , TBLW ,dar şi a altora.

exemplu: RPTK 0E5h

RSXM	resetează modul de extensie a semnului

sintaxă: [etichetă] RSXM [;com]
operanzi: nici unul

cuvinte: 1
execuţie: (PC)+1 → PC
0 → bitul SXM din registrul de stare ST1
Influenţează SXM .
descriere: Bitul modului de extensie al semnului din registrul de stare ST1 este facut 0.Acest fapt determină suprimarea extinderii semnului la deplasarea valorilor din memoria de date folosite de către următoarele instrucţiuni aritmetice : ADD, ADDT, ADLK, LAC, LALK, SBLK, SUB si SUBT .
exemplu: RSXM ;bitul SXM este facut 0 logic , fapt ce inhiba astfel extinderea
;semnului operanzilor instrucţiunilor următoare.

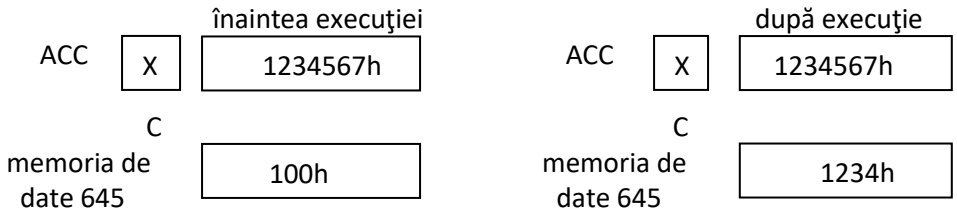
SACH	memorează cuvântul superior al acumulatorului , cu deplasare

sintaxă: ad.dir.: [etichetă] SACH dma [,deplasare] [;com]
ad.ind.: [etichetă] SACH {ind}[,deplasare][,ARP următor] [;com]
operanzi: 0 ≤ dma ≤ 127
0 ≤ ARP următor ≤ 7
0 ≤ deplasare ≤ 7(implicit = 0)

cuvinte: 1
execuţie: (PC)+1 → PC;
cei mai semnificativi 16 biti ai ((ACC)X $2^{deplasare}$) → dma
Nu este influenţată de SXM .

descriere: Instrucţiunea SACH copiază întregul acumulator într-un registru de deplasare în care este deplasat la stânga întregul număr de 32 de biţi de la 0 până la 7 biţi , după cum specifică codul de deplasare . Cei mai semnificativi 16 biti ai valorii deplasate sunt copiaţi apoi în locaţia de memorie de date specificată. Acumulatorul rămân nemodificat .

exemplu: SACH 5 , 4 ;(DP=5)
sau
SACH * ,4 ; dacă registrul auxiliar curent conţine 645;



SACL	memorează cuvântul inferior al acumulatorului , cu deplasare

sintaxă: ad.dir.: [etichetă] SACL dma [,deplasare] [;com]
ad.ind.: [etichetă] SACL {ind}[,deplasare][,ARP următor] [;com]
operanzi: 0 ≤ dma ≤ 127
0 ≤ ARP următor ≤ 7
0 ≤ deplasare ≤ 7(implicit = 0)

cuvinte: 1
execuţie: (PC)+1 → PC;
cei mai puţin semnificativi 16 biti ai ((ACC)X $2^{deplasare}$) → dma
Nu este influenţată de SXM .
descriere: Jumătatea inferioară a acumulatorului este deplasată la stanga de la 0 pana la 7 biţi, după cum specifică codul de deplasare , şi memorată apoi în locaţia de memorie de date specificată.Biţii cei mai puţin semnificativi sunt făcuţi 0, iar biţii cei mai semnificativi sunt pierduţi . Acumulatorul rămâne nemodificat.
exemplu: SACL 5 , 4 ;(DP=5)
sau
SACL * ,4 ; dacă registrul auxiliar curent conţine 645;

	înaintea execuţiei		după execuţie	
ACC	X	1234567h	X	1234567h
C				
memoria de date 645		100h		5670h

SFL	deplasează acumulatorul la stânga

sintaxă: [etichetă] SFL [;com]
operanzi: nici unul
cuvinte: 1
execuţie: (PC)+1 → PC
(ACC(31)) → C
(ACC(30-0)) → ACC(31-1)
0 → (ACC(0))
Influenţează C .
Nu este influenţată de SXM.
descriere: Instrucţiunea SFL deplasează întregul conţinut al acumulatorului la stânga cu un bit. Cel mai puţin semnificativ bit este făcut 0, iar bitul cel mai semnificativ este deplasat în bitul de transport (C). Trebuie remarcat ca SFL , spre deosebire de SFR , nu este influenţata de SXM .
exemplu: SFL

	înaintea execuţiei		după execuţie	
ACC	X	0AAAAAAA2h	1	55555544h
C				

SFR	deplasează acumulatorul la dreapta

sintaxă: [etichetă] SFR [;com]
operanzi: nici unul
cuvinte: 1

execuţie: (PC)+1 → PC
Dacă SXM=0, atunci
(ACC(0)) → C
(ACC(31-0)) → ACC(30-1)
0 → (ACC(31))

Dacă SXM=1, atunci
(ACC(0)) → C
(ACC(31-1)) → ACC(30-1)
(ACC(31)) → (ACC(31))
Influenţează C .
Este influenţată de SXM.

descriere: Instrucţiunea SFR deplasează acumulatorul la dreapta cu un bit.
Dacă SXM=1, instrucţiunea produce o deplasare aritmetică la dreapta. Bitul de semn, cel mai semnificativ bit, rămâne neschimbat şi este copiat şi în bitul 30.Bitul 0 este deplasat în bitul de transport(C) .
Dacă SXM=0, instrucţiunea produce o deplasare logică la dreapta.Toţi biţii acumulatorului sunt deplasaţi la dreapta cu un bit, cel mai semnificativ bit este făcut 0, iar bitul cel mai puţin smenificativ este deplasat în bitul de transport(C) .

exemplu: SFR ;(SXM=0)

	înaintea execuţiei		după execuţie	
ACC	X	0AAAAAAA2h	0	55555551h
C			C	

SFR ;(SXM=1)

	înaintea execuţiei		după execuţie	
ACC	X	0AAAAAAA2h	0	0D55555551h
C			C	

SOVM	setează modul depăşire

sintaxă: [etichetă] SFR [;com]
operanzi: nici unul
cuvinte: 1
execuţie: (PC)+1 → PC
1 → bitul OVM din registrul de stare ST0
Influenţează OVM .
descriere: Bitul modului de depaşire (OVM) din registrul de stare ST0 este făcut 1 , fapt ce stabileşte modul depaşire (cu saturare) .Dacă apare o depaşire când OBM=1, semaforul de depăşire (OV) se face 1 şi acumulatorul este încărcat cu cea mai mare valoare reprezentabilă pozitivă(7FFFFFFFh) sau negativă (80000000h) , în funcţie de sensul depăşirii.
OVM mai poate fi încărcat şi cu instrucţiunile LST şi ROVM.
exemplu: SOVM ;bitul OVM este făcut 1 logic , fapt ce determina stabilirea modului
;depăşire pentru toate operaţiile aritmetice următoare

SPM	setează modul de deplasare la ieşire din registrul P

sintaxă: [etichetă] SPM constantă [;com]
operanzi: 0 ≤ constantă ≤ 3
cuvinte: 1
execuţie: (PC)+1 → PC;
constantă → PM biţii modului de deplasare

Influenţează PM .

descriere: Ultimii doi biţi din codul instrucţiunii sunt copiaţi în câmpul PM din registrul de stare ST1. Biţii de stare PM controlează registrul de deplasare a ieşirii registrului P astfel :

1)dacă PM=0 (00 binar) , ieşirea registrului P nu este deplasată;

2)dacă PM=1 (01 binar), ieşirea registrului P este deplasată la stânga cu o poziţie şi completată cu zero;

3)dacă PM=2 (10 binar), ieşirea registrului P este deplasată la stânga cu patru pozitii şi completată cu zero;

3)dacă PM=3 (11 binar), ieşirea registrului P este deplasată la dreapta cu şase poziţii , i se extinde semnul şi i se pierd biţii cei mai puţin semnificativi .

Deplasările la stanga permit alinierea produsului pentru aritmetică fractionară . Deplasarea la dreapta cu 6 poziţii a fost inclusă pentru a permite realizarea a până la 128 de multiplicari-acumulari cu evitarea posibilitaţii de apariţie a depăşirii . PM mai poate fi încărcat şi cu instrucţiunea LST1.

exemplu: SPM 3 ;este selectat modul 3 de deplasare a produsului

SST	memorează registrul de stare ST0

sintaxă: ad.dir.: [etichetă] SST dma [;com]

ad.ind.: [etichetă] SST {ind}[,ARP următor] [;com]

operanzi: $0 \leq$ dma ≤ 127

$0 \leq$ ARP următor ≤ 7

cuvinte: 1

execuţie: (PC)+1 $\rightarrow$ PC;

(registrul de stare ST0) $\rightarrow$ dma

descriere: Registrul de stare ST0 este memorat în locaţia de memorie de date adresată . În modul de adresare direct , registrul de stare ST0 este întotdeauna memorat în pagina 0, indiferent de valoarea registrului DP . Procesorul fortează automat pagina 0 ca pagina curentă , iar specificarea locaţiei din pagina curentă se face în instrucţiune . Aceasta permite memorarea directa a registrului DP în memoria de date (la întreruperi, de exemplu) fără a trebui modificat el însuşi . În modul de adresare indirect , adresa memoriei de date este obţinută din registrul auxiliar selectat .

Instrucţiunea SST poate fi utilizată pentru a memora registrul de stare ST0 după întreruperi şi apeluri de subrutine . ST0 conţine biţii de stare:OV (semafor de depăşire) , OVM (bitul modului de depasire) ,INTM (semaforul modului de intrerupere) ,ARP (pointerul la registrul auxiliar) şi DP (pointerul la pagina de memorie date). Aceşti biti sunt memoraţi în cuvântul de memorie de date astfel:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARP			OV	OVM	1	INTM	DP								

Este de reţinut ca SST* poate fi folosită pentru memorarea registrului ST0 oriunde în memoria de date , in timp ce SST la adresare directă este fortată în pagina 0.

exemplu: SST 126 ;(DP nu contează)

sau

SST * ; dacă registrul auxiliar curent conţine 126;

	înaintea execuţiei		după execuţie
ST0	0E700h	ST0	0E700h
memoria de date 126	5491h	memoria de date 126	0E700h

SST1	memorează registrul de stare ST1

sintaxă: ad.dir.: [etichetă] SST dma [;com]
ad.ind.: [etichetă] SST {ind}[,ARP următor] [;com]
operanzi: 0 ≤ dma ≤ 127
0 ≤ ARP următor ≤ 7
cuvinte: 1
execuţie: (PC)+1 → PC;
(registrul de stare ST0) → dma
descriere: Registrul de stare ST1 este memorat în locaţia de memorie de date adresată .
În modul de adresare direct , registrul de stare ST1 este întotdeauna memorat în pagina 0, indiferent de valoarea registrului DP . Procesorul fortează automat pagina 0 ca pagina curentă , iar specificarea locaţiei din pagina curentă se face in instrucţiune . Aceasta permite memorarea directă a registrului DP în memoria de date (la întreruperi, de exemplu) fară a trebui modificat el însuşi . În modul de adresare indirect , adresa memoriei de date este obţinută din registrul auxiliar selectat .
Instrucţiunea SST1 poate fi utilizată pentru a memora registrul de stare ST1 după întreruperi şi apeluri de subrutine . ST1 conţine biţii de stare: ARB (bufferul pointerului la registrul auxiliar) , CNF (bitul de control al configuraţiei memoriei RAM de pe cip) , TC (semaforul test/control) , SXM (modul de extensie al semnului) , XF (bitul de stare a pinului XF), FO (formatul portului serial) , TXM (mod de transmisie) şi PM (modul de deplasare a registrului podus). Pe TMS320C25 ,ST1 mai conţine şi biţii de stare: C(transport (carry)) , HM (mod hold) şi FSM (mod de sincronizare a cadrului) . Aceşti biţi sunt memoraţi în cuvântul de memorie de date astfel :

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARB			CNF	TC	SXM	C	1	1	HM	FSM	XF	FO	TXM	PM	

Notă: La TMS320C26 , biţii 12 şi 7 conţin biţii CNF0 şi,respectiv , CNF1 (pentru semnificaţia lor, vezi instrucţiunea CONF).
Este de reţinut că SST1* poate fi folosită pentru memorarea registrului ST1 oriunde în memoria de date , în timp ce SST1 la adresare directă este forţată în pagina 0.

exemplu: SST1 127 ;(DP nu contează)
sau
SST 1 * ; dacă registrul auxiliar curent conţine 127;

	înaintea execuţiei		după execuţie
ST1	3180h	ST1	3180h
memoria de date 127	5491h	memoria de date 127	3180h

SSXM	setează modul de extensie a semnului

sintaxă: [etichetă] SSXM [;com]
operanzi: nici unul
cuvinte: 1
execuţie: (PC)+1 → PC
1 → bitul SXM din registrul de stare ST1
Influenţează SXM .
descriere: Bitul modului de extensie a semnului din registrul de stare ST1 este făcut 1. Acest fapt

determină extinderea semnului la deplasarea valorilor din memoria de date folosite de către următoarele instrucţiuni aritmetice: ADD, ADDT, ADLK, LAC, LALK, SBLK, SUB şi SUBT .
SXM mai poate fi încarcat şi cu instrucţiunile LST1 şi RSXM.

exemplu: SSXM ;bitul SXM este făcut 1 logic , permitând astfel extinderea
;semnului operanzilor instrucţiunilor următoare .

SUB	scade din acumulator cu deplasare

sintaxă: ad.dir.: [etichetă] SUB dma [,deplasare] [;com]
ad.ind.: [etichetă] SUB {ind}[,deplasare][,ARP următor] [;com]

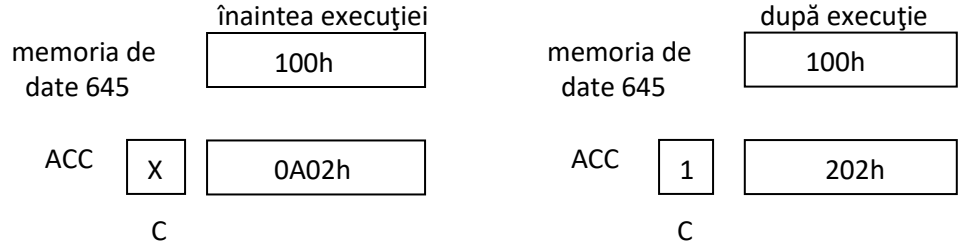
operanzi: $0 \leq$ dma ≤ 127
$0 \leq$ ARP următor ≤ 7
$0 \leq$ deplasare ≤ 15(implicit = 0)

cuvinte: 1

execuţie: (PC)+1 → PC;
(ACC) – [(dma) x $2^{deplasare}$] → ACC
Dacă SXM=1 , atunci (dma) va fi cu semn extins .
Dacă SXM=0 , atunci (dma) nu va fi cu semn extins .
Influenţează OV; este influenţată de OVM şi SXM.
Influenţează C.

descriere: Conţinutul locaţiei de memeorie adresate este deplasat la stânga şi scăzut din acumulator. În timpul deplasării, biţii inferiori se completează cu zero. Biţii superiori sunt completaţi cu valoarea bitului de semn ,dacă SXM =1, sau cu zero,dacă SXM=0.Rezultatul este memorat în acumulator.
Bitul C este făcut 0 dacă rezultatul scăderii generează un împrumut; astfel, C este făcut 1.

exemplu: SUB 5 , 3 ;(DP=5)
sau
SUB * ,3 ; dacă registrul auxiliar curent conţine 645;



SUBK	scade din acumulator un octet, imediat

sintaxă: [etichetă] SUBK constantă [;com]

operanzi: $0 \leq$ constantă ≤ 255

cuvinte: 1

execuţie: (PC)+1 → PC;
(ACC) – constanta pozitivă de 8 biţi → ACC
Influenţează OV şi C;este influenţată de OVM .
Nu este influenţată de SXM.

descriere: Valoarea imediată pe 8 biţi , aliniată la dreapta, este scăzută din acumulator şi rezultatul înlocuieşte conţinutul acumulatorului.Valoarea imediată este tratată ca un număr pozitiv pe 8 biţi, indiferent de valoarea lui SXM.

Bitul C este făcut 0 dacă rezultatul scăderii generează un împrumut; astfel,C este făcut 1.

exemplu: SUBK 0Dh

	înaintea execuţiei		după execuţie	
ACC	X	2195183h	ACC	0	2195176h
	C			C	

TBLR	citeşte tabel

sintaxă: ad.dir.: [etichetă] TBLR dma [;com]
ad.ind.: [etichetă] TBLR {ind}[,ARP următor] [;com]

operanzi: $0 \leq$ dma ≤ 127
$0 \leq$ ARP următor ≤ 7

cuvinte: 1

execuţie: (PC)+1 → PC;
(PFC) → (MCS)
(ACC(15-0)) → PFC
Dacă (contor de repetare) ≠ 0, atunci
(pma,adresat de PFC) → dma,
Modifică AR(ARP) şi ARP conform specificaţiei,
(PFC) + 1 → PFC,
(contor de repetare) -1 → contor de repetare;
În caz contrar ,
(pma,adresat de PFC) → dma,
modifică AR(ARP) şi ARP conform sfecificaţiei,
(MCS) → PFC.

descriere: Instrucţiunea TBLR transferă un cuvânt dintr-o locaţie din memoria de program într-o locaţie din memoria de date specificată în instrucţiune.Adresa din memoria de program este specificată de cuvântul inferior al acumulatorului(biţii 15-0). Pentru aceasta operaţie se realizează o citire din memoria program, urmată de o scriere în memoria de date.În modul repetat, TBLR devine efectiv instrucţiune într-un singur ciclu, iar contorul de preîncarcare(PFC) , devenit numărator al memoriei de program şi încărcat iniţial cu valoarea ACCL , este incrementat cu unu în fiecare ciclu.

exemplu: TBLR 5 ;(DP =5,ACC=102h)
sau
TBLR * ; dacă registrul auxiliar curent conţine 645;

	înaintea execuţiei		după execuţie	
ACC		102h	ACC	102h
memoria de program 102h		100h	memoria de program 102h	100h
memoria de date 645		0C4Fh	memoria de date 645	100h

TBLW	scrie tabelul

sintaxă: ad.dir.: [etichetă] TBLW dma [;com]
ad.ind.: [etichetă] TBLW {ind}[,ARP următor] [;com]

operanzi: $0 \leq$ dma ≤ 127
$0 \leq$ ARP următor ≤ 7

cuvinte: 1

execuţie: (PC)+1 → PC;
(PFC) → (MCS)
(ACC(15-0)) → PFC
Dacă (contor de repetare) ≠ 0, atunci
(dma) → pma,adresat în PFC
Modifică AR(ARP) şi ARP conform sfecificaţiei,
(PFC) + 1 → PFC,
(contor de repetare) -1 → contor de repetare;
În caz contrar ,
(dma) → pma,adresat de PFC
modifică AR(ARP) şi ARP conform sfecificaţiei,
(MCS) → PFC.

descriere: Instrucţiunea TBLW transferă un cuvant dintr-o locaţie din memoria de date într-o locaţie din memoria de program. Adresa memoriei de date este specificată de instrucţiune, iar adresa memoriei de program este specificată de cuvântul inferior al acumulatorului (bitii 15-0).Pentru această operaţie se realizează o citire din memoria de date ,urmată de o scriere în memoria de program.În modul repetat, TBLW devine efectiv instrucţiune într-un singur ciclu, iar contorul de preîncarcare(PFC) ,devine numărator al memoriei de program şi încarcat iniţial cu valoarea ACCL , este incrementat cu unu în fiecare ciclu.

exemplu: TBLW 5 ;(DP =5,ACC=102h)
sau
TBLW * ; dacă registrul auxiliar curent conţine 645;

	înaintea execuţiei		după execuţie
ACC	102h	ACC	102h
memoria de date 645	0C4Fh	memoria de date 645	0C4Fh
memoria de program 102h	100h	memoria de program 102h	0C4Fh

XOR — SAU EXCLUSIV cu acumulatorul

sintaxă: ad.dir.: [etichetă] XOR dma [;com]
ad.ind.: [etichetă] XOR {ind}[,ARP următor] [;com]

operanzi: $0 \leq$ dma ≤ 127
$0 \leq$ ARP următor ≤ 7

cuvinte: 1

execuţie: (PC)+1 → PC;
(ACC(15-0)) ⊕ (dma) → ACC(15-0)

(ACC(31-16)) → ACC(31-16)

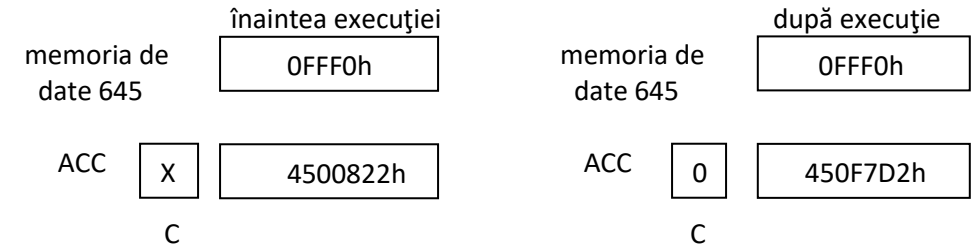
Nu este influenţată de SXM.

descriere: Jumatatea inferioară a acumulatorului este supusă unei operaţii SAU EXCLUSIV cu conţinutul locaţiei de memorie de date adresate. Jumatatea superioară a acumulatorului este neschimbată după execuţia acestei instrucţiuni .

exemplu: XOR 5 ;(DP =5)

sau

XOR * ; dacă registrul auxiliar curent conţine 645;



XORK	SAU EXCLUSIV imediat cu acumulatorul, cu deplasare

sintaxă: [etichetă] XORK constantă[,deplasare] [;com]

operanzi: constantă pe 16 biti

0 ≤ deplasare ≤ 15(implicit = 0)

cuvinte: 2

execuţie: (PC)+2 → PC;

(ACC(30-0)) ⊕ [constanta x $2^{deplasare}$] → ACC(30-0)

ACC(31) → ACC(31).

Nu este influenţată de SXM.

descriere: Constanta imediată de 16 biţi este deplasată la stânga şi supusă unei operaţii SAU EXCLUSIV cu acumulatorul.Biţii inferiori de sub şi biţii superiori de deasupra valorii deplasate sunt trataţi ca zerouri şi nu modifică,astfel,biţii corespunzatori ai acumulatorului.Este de remarcat că cel mai semnificativ bit al acumulatorului nu este modificat,indiferent de valoarea deplasării.

exemplu: XORK 0F00Fh,12

ZAC	zero în acumulator

sintaxă: [etichetă] ZAC [;com]

operanzi: nici unul

cuvinte: 1

execuţie: (PC)+1 → PC

0→ ACC

descriere: Conţinutul acumulatorului este înlocuit cu zero.Instrucţiunea ZAC este un caz particular al instrucţiunii LACK(are acelaşi cod ca şi LACK 0) .

exemplu: ZAC