SUPORT SEMINAR 7 - ASSEMBLER

PRELUCRĂRI în VIRGULĂ MOBILĂ

Variabile şi constante reale

Tipurile de date de tip real sunt:

Tip	Nr. biţi	Cifre semnificative	Interval de valori	Declarare
Short real	32	6-7	$1.18 * 10^{-38} \rightarrow 3.40 * 10^{38}$	vb DD 1.2
Long real	64	15-16	$2.23 * 10^{-308} \rightarrow 1.79 * 10^{308}$	vb DQ 2.3
10-byte real	80	19	$3.37 * 10^{-4932} \rightarrow 1.18 * 10^{4932}$	vb DT 1.8

ATENŢIE ! La declararea variabilelor reale se utilizează . şi nu ,

Declararea valorilor reale se poate realiza atât în format zecimal cât şi hexazecimal. În format zecimal se utilizează reprezentarea:

[+ | -] parte întreagă [parte zecimală] [E [+ | -] exponent]

Numerele sunt considerate a fi în bază zece. De exemplu:

```
a       DD      11.765          ;forma zecimală
b       DD      1.1765E+1       ;forma exponenţială zecimală
c       DQ      1176.5E-2       ;forma exponenţială zecimală
```

In format hexazecimal reprezentarea numerelor reale utilizează cifrele 0 → 9 şi A → F. Numărul trebuie să înceapă obligatoriu cu una din cifrele 0→9 şi să se termine cu eticheta **r** ce indică tipul numărului ca fiind real. Numărul de simboluri din reprezentarea hexazecimală este 8 pentru Short real, 16 pentru Long real şi 20 pentru 10-Byte real. În caz că primul simbol din număr este unul din caracterele A → F atunci se mai pune un 0 în faţă, iar dimensiunea ca număr de simboluri creşte cu unu. De exemplu:

```
a       DD      3F800000r
b       DQ      3D60000000000000r
c       DT      0A456C000000000000000r
```

Pentru a uşura lucrul cu date de tip real se pot defini propriile tipuri de date:

```
float       TYPEDEF     DD
double      TYPEDEF     DQ
long_double     TYPEDEF     DT
```

Formatul intern al numerelor reale este:

- pentru short real:

31	30 23	22 0
semn	exponent	parte zecimală

- pentru long real:

63	62 52	51 32	31 0
semn	exponent	parte zecimală	parte zecimală

- pentru real pe 10 octeți:

79	78 64	63	62 32	31 0
semn	exponent	întreg	parte zecimală	parte zecimală

Formele de reprezentare reprezintă formatul IEEE.
După cum se observă la numerele reale pe 10 octeți, partea întreagă este reprezentată de bitul 63. Acest bit permite atingerea unei precizii de 19 cifre. Partea întreagă este întotdeauna 1 la numere short real şi long real şi de aceea nu mai este memorată.
Valoarea exponențială reprezintă o putere a lui 2^n. Pentru a se reține şi valori negative de tipul 2^{-8}, valoarea exponentului este ajustată adunând la ea valoarea 127 pentru numere reale simplă precizie (short real), 1023 pentru numere reale dublă precizie (long real) şi 16383 pentru precizie extinsă (real pe 10 octeți).
Valoarea de ajustare este scăzută la conversia inversă din format intern (binar) în format extern (zecimal).

Arhitectura coprocesorului matematic

- are propriile registre de date şi de control: 8 registre de date organizate sub formă de stivă şi 7 registre de control (asemenea registrelor flag);
- cele 8 registre de date sunt pe 80 de biți şi sunt organizate sub formă de stivă, cu toate că permit şi acces direct; datele sunt încărcate în registrul din vârf, ST(0) şi coboară către bază, registrul ST(7);
- registrul din vârful stivei este ST(0) sau ST, urmând apoi ST(1), ST(2), …, ST(7);
- odată încărcate datele sunt convertite automat la format pe 10 octeți;
- stiva registrelor coprocesorului este:

Registru	79	78 63	62 0
ST(0)			
ST(1)			
ST(2)			
ST(3)			
ST(4)			
ST(5)			
ST(6)			
ST(7)			

semn exponent parte zecimală

Tipuri de instrucțiuni

Cele 8 registre de date pot fi accesate ca elemente ale unei stive sau ca elemente individuale asemenea registrelor procesorului. Toate instrucțiunile coprocesorului pentru prelucrarea valorilor reale încep cu **F**.

*Format instrucțiune	Sintaxă	Operatori impliciți
Classical stack	**F**instrucțiune	**ST,ST(1)**
Memory	**F**instrucțiune **variabilă**	**ST**
Register	**F**instrucțiune **ST(nr),ST** **F**instrucțiune **ST,ST(nr)**	-
Register pop	**F**instrucțiune**P ST(nr),ST**	-

*- preluate din Micrsoft MASM Programmer's Guide

FORMATUL CLASSICAL STACK

- sinatxă **F**instrucțiune
- consideră registrele coprocesorului ca părți componente ale unei stive; valorile sunt adăugate sau scoase din vârful stivei, adică ST sau ST(0);
- operatorii impliciți sunt ST(0) (sursă) şi ST(1) (destinație);
- rezultatul instrucțiunii este pus în registrul destinație, iar valoarea din sursă (adică ST) este scoasă din stivă;
- **ATENȚIE !** Nu toate instrucțiunile de tip Classical Stack utilizează cei doi operanzi impliciți, ci doar acele instrucțiuni care în mod uzual au doi operanzi (de exemplu Fadd); de exemplu instrucțiuni de acest format sunt:

FLD1 – încarcă în ST valoarea 1;
FLDZ – încarcă în ST valoarea 0;
FLDPI – încarcă în ST valoarea lui pi, 3,14;
FLDL2E – încarcă în ST $\log_2 e$;
FLDL2T – încarcă în ST $\log_2 10$;
FLDLG2 – încarcă în ST $\log_{10} 2$;
FLDLN2 – încarcă în ST $\log_e 2$.

ATENȚIE ! Pentru a vizualiza registrele coprocesorului în Turbo Debugger se deschide fereastra Numeric Processor din **View → Numeric processor.**

ATENȚIE ! Instrucțiunea **Fxch** interschimbă valorile din ST şi ST(1) dar fără a scoate valoarea din ST**.**

ATENȚIE ! Dacă se dau mai mult de 8 instrucțiuni succesive de tip FLD (adică se încarcă mai mult de 8 valori în registrele ST, ST(1), …, ST(7) fără a se scoate nici una dintre ele, coprocesorul indică situația punând în registrul ST(0) valoarea NAN (Not a Number)**.**

Exemplu:

- evaluarea expresiei: e = a + 1 + b * pi

```
DateIn SEGMENT
     a DD 1.23
     b DD 3.6
DateIn ENDS

DateOut   SEGMENT
     e DD ?
DateOut ENDS

Main SEGMENT
     ASSUME CS:Main, DS:DateIn, ES:DateOut
start:
     mov AX,DateIn
     mov DS,AX
     mov AX,DateOut
     mov ES,AX

     Fld b      ;#1
     Fldpi      ;#2
     Fmul       ;#3
                ;are operanzi impliciți pe ST(1),ST
     Fld1       ;#4
     Fadd       ;#5
     Fld a      ;#6
     Fadd       ;#7
     Fstp es:e  ;#8
                ;pune rezultatul în e şi scoate valoarea din ST

     mov AX,4c00h
     int 21h
Main ENDS

end start
```

Pentru programul anterior stiva registrelor coprocesorului trece prin următoarele faze:

	#1	#2	#3	#4	#5	#6	#7	#8
ST	3.6	3.14	11.304	1	12.304	1.23	13.534	-
ST(1)	-	3.6	-	11.304	-	12.304	-	-
ST(2)	-	-	-	-	-	-	-	-

FORMATUL MEMORY

- sintaxă **F**instrucțiune **variabilă**
- instrucțiuni care realizează transfer de valori în şi din registrele coprocesorului; consideră registrele coprocesorului incluse într-o stivă;
- operator implicit registrul ST sau ST(0);
- pune sau scoate valori din vârful stivei; exemple de astfel de instrucțiuni:

FLD nume_variabilă – încarcă pe stivă (adică în ST) valoarea variabilei;
FST nume_variabilă – copiază (nu scoate) valoarea din vârful stivei în variabilă;

Exemplu:
- interschimbarea a 2 valori reale utilizând coprocesorul

```
DateIn SEGMENT
     a DD 1.23
     b DD 3.6
DateIn ENDS

Main SEGMENT
     ASSUME CS:Main, DS:DateIn
start:
     mov AX,DateIn
     mov DS,AX

     Fld a      ;#1
     Fld b      ;#2
     Fstp a     ;#3
                ;pune valoarea din ST în a şi o scoate
     Fstp b     ;#4

     mov AX,4c00h
     int 21h
Main ENDS

end start
```

Pentru programul anterior stiva registrelor coprocesorului trece prin următoarele faze:

	#1	#2	#3	#4
ST	1.23	3.6	1.23	-
ST(1)	-	1.23	-	-
ST(2)	-	-	-	-

FORMATUL REGISTER

- sintaxă **F**instrucțiune**P ST(nr),ST**;

- instrucțiuni care tratează registrele coprocesorului ca registre independente şi ca părți ale unei stive;
- operandul sursă trebuie să fie ST;
- primul operand este destinația, iar cel de-al doilea este sursa;
- rezultatul instrucțiunii este pus în destinație, iar valoarea din sursă (ST – vârful stivei) este scoasă din stivă;
- astfel de instrucțiuni sunt cele ce implementează operații matematice şi care necesită scoaterea valorii sursei din stivă:

Exemplu:

- evaluarea expresiei: $e = a + b + a*b + a^2$

```
DateIn SEGMENT
     a DD 1.23
     b DD 3.6
DateIn ENDS

DateOut   SEGMENT
     e DD ?
DateOut ENDS

Main SEGMENT
     ASSUME CS:Main, DS:DateIn, ES:DateOut
start:
     mov AX,DateIn
     mov DS,AX
     mov AX,DateOut
     mov ES,AX

     Fld a          ;#1
     Fld a          ;#2
     Fmul ST,ST     ;#3
     Fxch ST(1)     ;#4
     Fld b          ;#5
     Fadd ST(2),ST  ;#6
     Fmul ST,ST(1)  ;#7
     Fadd ST,ST(1)  ;#8
     Fadd ST,ST(2)  ;#9
     Ffree ST(2)    ;#10
                    ;eliberez registrul ST(2)
     Ffree ST(1)    ;#11
     Fstp ES:e      ;#12

     mov AX,4c00h
     int 21h
Main ENDS
```

```
end start
```

Pentru programul anterior stiva registrelor coprocesorului trece prin următoarele faze:

	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12
ST	1.23	1.23	1.5129	1.23	3.6	3.6	4.428	5.658	10.7709	10.7709	10.7709	-
ST(1)	-	1.23	1.23	1.5129	1.23	1.23	1.23	1.23	1.23	1.23	-	-
ST(2)	-	-	-	-	1.5129	5.1129	5.1129	5.1129	5.1129	-	-	-

FORMATUL REGISTER

- sintaxă **F**instrucțiune **ST(nr),ST** sau **F**instrucțiune **ST,ST(nr)**;
- instrucțiuni care tratează registrele coprocesorului ca registre independente şi nu ca părți ale unei stive;
- indiferent de sintaxă, unul dintre operanzi trebuie să fie ST;
- primul operand este destinația, iar cel de-al doilea este sursa;
- rezultatul instrucțiunii este pus în destinație, sursa rămânând nemodificată;
- astfel de instrucțiuni sunt cele ce implementează operații matematice şi care necesită scoaterea valorii din vârful stivei;

Exemplu:
- evaluarea expresiei: e = a + b + c + d

```
DateIn SEGMENT
     a DD 1.23
     b DD 3.6
     c DD 3.5
     d DD 7.43
DateIn ENDS

DateOut   SEGMENT
     e DD ?
DateOut ENDS

Main SEGMENT
     ASSUME CS:Main, DS:DateIn, ES:DateOut
start:
     mov AX,DateIn
     mov DS,AX
     mov AX,DateOut
     mov ES,AX

     Fld a          ;#1
     Fld b          ;#2
     Fld c          ;#3
     Fld d          ;#4
     Faddp ST(3),ST ;#5
```

```
      Faddp ST(2),ST ;#6
      Faddp ST(1),ST ;#7
      Fstp ES:e      ;#8

      mov AX,4c00h
      int 21h
Main ENDS

end start
```

Pentru programul anterior stiva registrelor coprocesorului trece prin următoarele faze:

	#1	#2	#3	#4	#5	#6	#7	#8
ST	1.23	3.6	3.5	7.43	3.5	3.6	15.76	-
ST(1)	-	1.23	3.6	3.5	3.6	12.16	-	-
ST(2)	-	-	1.23	3.6	8.66	-	-	-
ST(3)	-	-	-	1.23	-	-	-	-

Exemplu complet de program

- să se rezolve ecuația de gradul al 2-lea: $aX^2+bX+c=0$ unde a = 3.8, b = 8.2 şi c = 1.6.

```
DateIn SEGMENT
      a DD 3.8
      b DD 8.2
      c DD 1.6
DateIn ENDS

DateOut     SEGMENT
      x1 DD ?
      x2 DD ?
DateOut ENDS

Main SEGMENT
      ASSUME CS:Main, DS:DateIn
start:
      mov AX,DateIn
      mov DS,AX

      Fld1        ;incarc valoarea 1 in ST - #1
      Fadd ST,ST  ;adun ST cu ST si obtin 2 - #2
      FLD ST      ;incarc valoarea din ST - #3
      Fmul a      ;obtin 2*a - #4

      Fmul ST(1),ST     ;calculez valoarea lui 4*a - #5
      Fxch        ;interschimb ST cu ST(1) - #6
      Fmul c      ;inmultesc ST cu c si obtin valoarea lui 4*a*c - #7

      Fld b       ;incarc valoarea lui b in ST - #8
      Fmul ST,ST  ;inmultesc ST cu ST si obtin b*b - #9
      FsubR       ;scadere inversa (adica ST(1) = ST-ST(1) si scoate
                  ;valoarea din ST  - #10
```

```
                    ;doar Fsub ar fi scazut ST(1) = ST(1)-ST
                    ;am obtinut valoarea lui b*b - 4*a*c

;ATENȚIE PROGRAMUL NU VALIDEAZA DACA VALOAREA DIN CARE FACE RADICAL
;ESTE SAU NU MAI MICA DECAT ZERO (tema !!!)

      Fsqrt       ;radical din ST adica din b*b-4*a*c - #11
      Fld b       ;incarc valoarea lui b - #12
      Fchs        ;schim semnul valorii din ST - #13
      Fxch        ;interschimb valorile din ST si ST(1) - #14

      Fld ST      ;incarc valoarea lui ST (valoarea radicalului) - #15
      Fadd ST,ST(2)     ;adun in ST ST+ST(2) si obtin -b + valoare
                        ;radical - #16
      Fxch              ;interschimb valorile din ST si ST(1) - #17
      Fsubp ST(2),ST    ;fac ST(2) = ST(2) - ST si apoi scot valoarea
                        ;lui ST - #18

      ASSUME DS:DateOut       ;schimb segmentul de date
      mov AX,DateOut          ;pentru a scrie valoarea rezultatelor
      mov DS,AX

      Fdiv ST,ST(2)     ;fac ST = ST / ST(2) - #19
      Fstp x1           ;scot rezultatul in x1 - #20
      FdivR       ;impartire inversa adica ST(1) = ST /ST(1)
                  ;si scot valoarea din ST - #21
                  ;pe impartire normala Fdiv face ST(1) = ST(1)/ST
      Fstp x2     ;scot rezultatul in x2 - #22


      mov AX,4c00h
      int 21h
Main ENDS

end start
```

Pentru programul anterior stiva registrelor coprocesorului trece prin următoarele faze:

	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
ST	1	2	2	7.6	7.6	15.2	24.32	8.2	67.24	42.92
ST(1)	-		2	2	15.2	7.6	7.6	24.32	24.32	7.6
ST(2)	-	-	-	-	-	-	-	7.6	7.6	-
ST(3)	-	-	-	-	-	-	-	-	-	-
ST(4)	-	-	-	-	-	-	-	-	-	-

	#11	#12	#13	#14	#15	#16	#17	#18	#19	#20	#21
ST	6.55	8.2	-8.2	6.55	6.55	-1.64	6.55	-1.64	0.21	-14.75	-1.94
ST(1)	7.6	6.55	6.55	-8.2	6.55	6.55	-1.64	-14.75	-14.75	7.6	-
ST(2)	-	7.6	7.6	7.6	-8.2	-8.2	-8.2	7.6	7.6	-	-
ST(3)	-	-	-	-	7.6	7.6	7.6	-	-	-	-

ST(4)	-	-	-	-	-	-	-	-	-	-	-

Alte tipuri de instrucțiuni

ATENȚIE ! In cazul instrucțiunilor cu 2 operanzi , primul este destinația iar al doilea este sursa. Rezultatul operației se pune în destinație, deci aceste instrucțiuni sunt echivalente operației `destinație = destinație <operație> sursa`. In cazul instrucțiunilor care au un **R** la sfârşit (de ex: FsubR, FdivR) operația are loc invers, adică: `destinație = sursă <operație> destinație`.
Instrucțiunile care au un **P** la sfârşit scot valoarea din ST.

Operații aritmetice:

FADD [operanzi] FADDP op1,op2 FIADD vb	adună două numere reale adună cei doi operanzi şi descarcă stiva adună un întreg de 16/32 de biți la ST
FSUB [operanzi] FSUB op1,op2 FSUB vb FSUBR [operanzi] FSUBRPop1,op2 FSUBR vb	scade numere reale scade op2 din op1 şi se descarcă stiva scade întreg de 16/32 de biți din ST scade inversa numerele reale scade op1 din op2 şi se descarcă stiva scade ST dintr-un întreg de 16/32 de biți
FMUL [operanzi] FMULP op1,op2 FMUL vb	înmulțire numere reale înmulțire de numere reale (op1 * op2), cu descărcarea stivei înmulțire ST cu întreg din memorie, de 16/32 de biți
FDIV [operanzi] FDIVP op1,op2 FIDIV vb FDIVR [operanzi] FDIVRP opl/op2 FDIVR vb	împărțire de numere reale (op1/op2) împărțire op1/op2 şi descărcarea stivei împărțirea lui ST la un întreg de 16/32 de biți împărțire inversă de numere reale (op2/op1) împărțire inversă de numere reale (op2/op1) cu descărcarea stivei împărțirea unui întreg de 16/32 de biți la ST
FSQRT FSCALE FPREM FRNDINT FXTRACT FABS FCHS	rădăcină pătrată din ST ST←ST*$2^{ST(1)}$, ST(1) interpretat ca întreg împărțire modulo: ST / ST(1) rotunjire vârf stivă, ST, la întreg ' descompune numărul din vârful stivei în două numere: exponent şi mantisă; după care mantisa este depusă în stivă. ST = |ST| ST = -ST

Instrucțiuni matematice pentru calcule complexe

FCOS	cosinusul lui ST

FPATAN	arctangent de ST(1)/ŞT şi extrage (descarcă) ST
FPTAN	tangenta parţială de ST (Y/X - rezultat), Y înlocuieşte ST, X este depus în stivă
FSIN	sinusul lui ST
FSINCOS	(temp)← ST, ST = sin (temp) şi depune în stivă cos(temp)
F2XM1	2ST-1; -1≤ST≤1
FYL2X	ST(1) = ST(1) * $\log_2$(ST(0)) şi descarcă stiva
FYL2XP1	ST(1) = ST(1) * $\log_2$(ST(0)+1) şi descarcă stiva

Instrucţiuni de comparare şi de control

O parte din flag-urile celor 7 registre (pe 16 biţi) de control gestionează operaţiile coprocesorului şi starea curentă a acestuia.

Nr. registru	Registre de control	
1	Cuvânt de control	
2	Cuvânt de stare	
3	Cuvânt Eticheta (Tag)	
4		Instruction Pointer
5		
6		Operand pointer
7		

Registrul asociat cuvântului de stare este cel mai folosit, restul fiind utilizaţi de programatorii de sistem.
Modul în care sunt aranjaţi biţii din octetul superior este identic cu modul de aranjare a flag-urilor din registrul de flag al procesorului. Situaţia un este întâmplătoare şi facilitează controlul execuţiei programului în cazul coprocesorului (acesta nu are instrucţiuni de salt condiţionat şi atunci trebuie sa trimită procesorului informaţia).

Octetul superior al cuvântului de stare:

15	14	13	12	11	10	9	8
	C3				C2	C1	C0

Octetul inferior al registrului de flag-uri al procesorului:

7	6	5	4	3	2	1	0
SF	ZF		AF		PF		CF

Coprocesorul are instrucţiuni care setează flag-urile din cuvântul de stare (unul dintre registrele de stare). Acest registru este utilizat pentru a controla execuţia programului şi pentru a introduce structuri de control prin intermediul salturilor condiţionate. Singura problemă este că, coprocesorul matematic nu are instrucţiuni de salt, doar procesorul are.

Din acest motiv cuvântul de stare trebuie încărcat în memorie în Flag-urile procesorului astfel încât să se păstreze semnificaţia valorilor. Soluţia este dată de încărcarea octetului superior din cuvântul de stare în octetul inferior din registrul de flag-uri al procesorului prin secvenţa:

```
Fstsw   variabila_de_tip_word        ;pun valoarea cuvântului de stare într-o variabilă
Fwait
mov     AX, variabila_de_tip_word    ;încarc valoarea variabilei în AX
sahf                                 ;încarc valoarea din AH în registrul de Flag al
                                     ;procesorului
```

Odată încărcată această valoare se pot instrucţiuni de salt condiţionat pe baza valorilor din registrele coprocesorului matematic.

Coprocesorul are o serie de instrucţiuni care analizează valoare din registrul ST în relaţie cu valoarea altui operand şi raportează rezultatul într-un cod de condiţie (dat de biţii C0, C1, C2, C3) ai cuvântului de stare. Operaţiile de bază sunt de comparare şi de testare(comparare cu zero).
Modul în care sunt afectaţi biţii C0, C1, C2, C3 este:

După FCOM	**După FTEST**	**C3**	**C2**	**C0**
ST > sursă	ST > 0	0	0	0
ST < sursă	ST < 0	0	0	1
ST = sursa	ST = 0	1	0	0
Nu se poate compara	Nu se poate testa; ST are valoarea NAN	1	1	1

FCOM op	compară ST cu operandul (registru sau rnemorie)
FCOMP op	compară ST cu operandul (registru sau memorie) şi descarcă stiva
FCOMPP	compară ST cu ST(1) şi extrage ambele valori din stivă
FICOM vb	compară ST cu un întreg de 16/32 de biţi
FICOMP vb	compară ST cu un întreg de 16/32 de biţi şi extrage ST din stivă
FUCOM op	comparare cu NaN (Not a Number)
FUCOMP op	comparare cu NaN (Not a Number) şi extrage vârful stivei
FUCOMPP op	compară ST cu ST(1) şi extrage arabele valori din stivă
FTST	compară SI cu 0.0
FXAM	examinează ST şi stabileşte codurile de condiţie
FINIT	resetează coprocesorul şi cuvintele de stare şi control
FFREE [reg]	marchează registrul indicat ca fiind eliberat

ATENŢIE ! Pentru examen trebuie citit din Bibliografia indicată şi formatul BCD de reprezentare a valorilor întregi.