

LRN

Programmer en LMS

TI-58 / TI-58C / TI-59

Pierre Houbert

Introduction

Les calculatrices programmables Texas Instruments **TI58** et **TI59** sont apparues en 1977, suivies en 1979 par la **TI58C**.

Basées sur un système AOS (notation algébrique directe), elles étaient programmables grâce à un langage spécifique nommé LMS (langage machine spécialisé).

Certains utilisateurs ont plus vu dans ces machines leur côté « calculatrice scientifique » (ou mathématique) en raison de leurs nombreuses fonctions mathématiques et statistiques, d'autres ont adopté ces calculatrices comme des « ordinateurs de poche » et ont même inventé, dans ces années de micro-informatique naissante, le terme de « pico-informatique ».

Nous aborderons ici le côté calculatrice programmable et essayerons de découvrir ce langage, a priori rudimentaire et simpliste, qui a pourtant su fasciner de nombreux adeptes.

En effet, ce langage s'est avéré être réellement attractif car suffisamment complet pour élaborer des programmes complexes.

Le champ des applications possibles a même permis une utilisation professionnelle dans certains domaines.

Les modules de programmes commercialisés concernaient les mathématiques, la navigation, l'ingénierie électrique, l'agriculture, l'investissement financier, la gestion de stock et bien d'autres activités sans oublier les jeux.

Les seules limites étaient dues aux contraintes physiques de ces machines : pas d'affichage alphanumérique (mais impression

papier de textes), taille de mémoire, support de sauvegarde (cartes magnétiques pour la **TI59** seulement).

Alors pourquoi se pencher aujourd'hui, à l'ère des « smartphones » et autres « tablettes », sur ces machines ancestrales et ce langage d'autrefois ?

Pour la même raison qui fait qu'à l'âge des navettes spatiales, des trains à grande vitesse, et autres engins de vitesse, nos enfants, et nos petits-enfants continuent à vouloir apprendre à faire du vélo : pour le plaisir !

Aujourd'hui des émulateurs de TI existent sur divers systèmes d'exploitation (MS Dos, Windows, Android, Pocket PC) et permettent de retrouver ce plaisir particulier de programmer avec un tel langage.

Premier programme

Premiers pas

Pour commencer, observons le clavier de notre calculatrice.



La première touche que nous aborderons est la touche **2nd**. Elle va nous permettre d'accéder à la fonction « seconde » d'une touche, ainsi pour obtenir Π (pi) nous devons utiliser la fonction seconde de la touche **3**.

Donc,

la séquence de touche **2nd** **Π** donnera **3.14159265359**

Pour calculer le périmètre d'un cercle de 4 cm de rayon, il faut faire $4 \times 2 \times \Pi =$ et donc taper :

4 **×** **2** **×** **2nd** **Π** **=**

Nous pouvons faire un premier programme permettant de calculer le périmètre d'un cercle quelque soit son rayon...

Ce programme sera du genre :

- Saisie nombre
- Multiplier par 2
- Multiplier par Pi
- Affichage résultat

La saisie du nombre se fera au clavier puis il faudra lancer l'exécution du programme qui s'arrêtera en affichant le résultat.

Pour lancer le programme (et l'arrêter) nous utiliserons la touche (et l'instruction) **R/S** qui signifie *Run / Stop*.

Notre programme sera donc de la forme :

× **2** **×** **2nd** **Π** **=** **R/S**

Introduction du programme

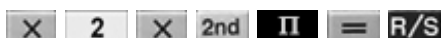
Pour entrer un programme, il faut passer en mode programmation à l'aide de la touche **LRN** (*Learn*).

Lorsque nous appuyons sur cette touche **LRN**, l'affichage change et fait apparaître deux groupes de chiffres séparés par un espace.



Le premier groupe, composé de 3 chiffres représente l'adresse de l'instruction (nous parlerons plutôt de **pas** de programme), et le deuxième groupe, composé de 2 chiffres, représente le code de l'instruction.

A chaque instruction représentée sur le clavier correspond un code de deux chiffres (de 00 à 99) ainsi notre programme



pourrait s'écrire

65 02 65 89 95 91

puisque les codes respectifs sont

- **65** pour
- **02** pour
- **65** pour
- **89** pour
- **95** pour
- **91** pour

Pour repasser en mode « calculatrice », ou sortir du mode programmation, nous appuyons sur **LRN**.

Avant d'introduire notre programme, nous allons nous assurer qu'aucun autre programme n'est en mémoire en effaçant la mémoire programme avec **CP** obtenu avec **2nd** **CP** (*Clear Program*).

Si nous repassons en mode programmation par appui sur **LRN**, nous sommes sur le pas 000 avec 00 comme code d'instruction.

Appuyons sur **X**, le pas 001 apparaît avec 00 comme code d'instruction.

Appuyons ensuite sur **2**, puis **X**, puis **2nd** **II**, puis **=**, puis **R/S**.

Au fur et à mesure de notre saisie, nous pouvons voir les pas de programme s'incrémenter pour un positionnement sur le pas de l'instruction suivante à introduire.

Pour vérifier notre saisie, nous avons deux solutions : soit nous « promener » dans notre programme pour afficher les pas successifs, soit sortir du mode programmation (avec **LRN**) et imprimer notre programme.

Promenons nous...

Pour vérifier notre saisie, nous pouvons « remonter » dans notre programme à l'aide de **BS↑**.

Chaque appui sur **BS↑** nous fait « remonter » d'un pas et nous voyons s'afficher l'adresse du pas et le code de l'instruction :

BS↑ affiche **005 91** puis **BS↑ 004 95** puis **BS↑ 003 89** puis **BS↑ 002 65** puis **BS↑ 001 02** puis **BS↑ 000 65**.

Nous pouvons aussi « redescendre » dans notre programme à l'aide de **SS↓**.

SS↓ affiche **001 02**, puis **SS↓ 002 65**, puis **SS↓ 003 89** puis **SS↓ 004 95** puis **SS↓ 005 91**.

Appuyons sur **LRN** pour revenir en mode « calculatrice ».

Nous avons quitté le mode programmation alors que le pointeur de pas était sur le pas **005**.

Si nous ré-appuyons sur **LRN** pour repasser en mode programmation, c'est le pas **005** qui s'affiche.

Si nous tentions de lancer l'exécution du programme, rien ne se passerait car le pointeur d'exécution est positionné sur l'instruction d'arrêt.

Pour revenir, en mode « calculatrice », il faut appuyer sur **LRN** puis **RST** (*Reset*) pour ramener le pointer sur le pas **000**.

Pour vérification du programme, nous allons l'imprimer en utilisant **LST** (**2nd** **List**)

Sur l'imprimante, nous obtenons :

000	65	x
001	02	2
002	65	x
003	89	π
004	95	=
005	91	R/S

L'impression papier nous donne l'adresse (pas) de l'instruction, son code et aussi sa traduction.

Premier test

Nous pouvons maintenant tester notre programme.

Il faut :

- ramener le pointeur à **000** : **RST**
- saisir un rayon : par exemple **2** **5**
- lancer le programme : **R/S**

et nous obtenons 157,0796327

L'utilisation pourrait être améliorée en évitant d'avoir à utiliser des touches telles que **RST** et **R/S**.

En effet, la calculatrice possède des touches de « fonction » (**A**, **B**, **C**, **D**, **E**) qui pourraient nous être utiles.

Nous allons donc utiliser la notion de « label » (ou étiquette).

Pour modifier notre programme, nous ramenons le pointeur à l'adresse **000** avec **RST**, puis basculons en mode programmation avec **LRN**.

Nous sommes sur le pas **000** devant lequel nous allons insérer 2 lignes en utilisant **INS** deux fois : **2nd** **Ins** **2nd** **Ins**

Nous pouvons maintenant mettre notre déclaration de label avec **2nd** **Lbl** (**LBL**), puis **A**.

Nous revenons en mode « calculatrice » pour imprimer : **LRN**, puis **RST** **2nd** **List**

Sur l'imprimante, nous obtenons :

000	76	LBL
001	11	A
002	65	X
003	02	2
004	65	X
005	89	π
006	95	=
007	91	R/S

Nous pouvons maintenant re-tester notre programme.

Il faut :

- saisir un rayon : par exemple **2** **5**
- lancer le programme : **A**

et nous obtenons 157,0796327

Si après l'exécution nous appuyons sur **LRN** pour passer en mode programmation, nous nous apercevons que le pointeur d'exécution est placé sur le pas **008**.

Nous allons ajouter une deuxième partie permettant le calcul de la surface du cercle :

2nd **Lbl** **B** **X²** **X** **2nd** **π** **=** **R/S**

soit : **LBL B X² X π = R/S**

puis **LRN** pour retourner en mode « calculatrice ».

RST **2nd** **List** pour imprimer.

000	76	LBL
001	11	R
002	65	x
003	02	2
004	65	x
005	89	π
006	95	=
007	91	R/S
008	76	LBL
009	12	B
010	33	x^2
011	65	x
012	89	π
013	95	=
014	91	R/S

Maintenant un nombre n suivi de **A** affiche le périmètre et un nombre n suivi de **B** affiche la surface d'un cercle.

Nous pouvons maintenant modifier ce programme afin de ne saisir le rayon qu'une seule fois et faire nos deux calculs à la suite en tapant :

rayon **A** **B**

Pour le faire, nous allons devoir stocker en mémoire le rayon dans la procédure **A** et rappeler l'information stockée dans la procédure **B**.

Stockage en mémoire

La TI contient plusieurs « zones » de stockage pour conserver les données utilisées. Ces zones sont appelées **Registres**.

Le premier registre est celui qui correspond à l'affichage numérique : c'est le registre « **x** ».

Le second registre est le registre de test nommé « **t** ».

La commande **↔** permet, comme son nom l'indique « *x échange t* », d'échanger les valeurs de **x** et **t**.

Exemple :

1 2 3 **↔** 456 met la valeur 123 dans **t** et 456 dans **x**
↔ échange : 123 dans **x** et 456 dans **t**

D'autres registres sont utilisés pour le stockage, ils sont numérotés de 00 à 99 ¹.

Pour manipuler ces registres différentes instructions sont utilisables :

STO nn copie le registre **x** dans le registre nn

RCL nn copie le registre nn dans le registre **x**

SUM nn ajoute le registre **x** au registre nn

INV SUM nn soustrait le registre **x** du registre nn

2nd Prd nn multiplie le registre nn par le registre **x**

INV 2nd Prd nn multiplie le registre nn par le registre **x**

2nd Exc nn échange le registre nn avec le registre **x**

¹ Diffère selon le modèle de TI et les options retenues – Voir OP 16 et OP 17

Pour notre programme de « cercle », nous allons donc stocker le rayon dans le registre 01 pour le reprendre ensuite.

Derrière **2nd** **Lbl** **A** nous allons insérer **STO** **0** **1** .

Lors de la saisie de l'adresse du registre (01), l'affichage n'avance pas de deux pas.

En effet, après **STO**, 2 caractères sont attendus et ne prennent qu'un seul pas de programme.

Derrière **2nd** **Lbl** **B** nous insérons **RCL** **0** **1** .

Nous obtenons :

```

000 76 LBL
001 11 A
002 42 STO
003 01 01
004 65 x
005 02 2
006 65 x
007 89 PI
008 95 =
009 91 R/S
010 76 LBL
011 12 B
012 43 RCL
013 01 01
014 33 X2
015 65 x
016 89 PI
017 95 =
018 91 R/S

```

Pour tester, il suffit de saisir le rayon, d'appuyer sur **A** pour obtenir le périmètre puis sur **B** pour obtenir la surface.

Si en mode « calculatrice » nous appuyons sur **RCL** **0** **1** , le rayon s'affiche.

Impression

Nous allons nous servir de l'imprimante afin d'améliorer la présentation des résultats.

Pour l'utilisation de l'imprimante, nous avons déjà vu **LST** (**2nd** **List**) qui permet de lister un programme.

Nous pouvons aussi utiliser :

INV **2nd** **List** pour imprimer le contenu des registres (**INV LST**)

2nd **Prt** pour imprimer le registre **x** (**PRT**)

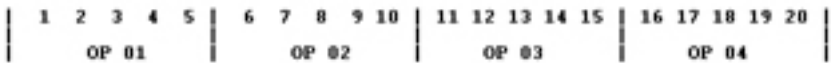
2nd **Adv** pour avancer d'une ligne (**ADV**)

De plus, des fonctions « spéciales » sont accessibles grâce à l'instruction **OP** (**2nd** **Op**) :

- **OP 01**, **OP 02**, **OP 03**, **OP 04** et **OP 05** permettent d'imprimer un texte alphanumérique jusqu'à 20 caractères, une ligne d'imprimante faisant 20 caractères de large.
- **OP 06** imprime le registre **x** suivi de 5 caractères alphanumériques
- **OP 07** imprime une courbe avec le caractère « * »
- **OP 08** imprime la liste des étiquettes (labels) utilisées par le programme en mémoire.

L'impression d'un texte alphanumérique se fait sur une ligne de 20 caractères divisée en 4 groupes de 5 caractères.

- **OP 01** affecte le groupe 1 (extérieur gauche)
- **OP 02** affecte le groupe 2 (intérieur gauche)
- **OP 03** affecte le groupe 3 (intérieur droit)
- **OP 04** affecte le groupe 4 (extérieur droit)



- **OP 05** imprime la ligne
- **OP 00** efface le contenu des 4 groupes (zéro)

Pour affecter des valeurs aux groupes, la TI utilise une table de correspondance de caractères :

	:	0	1	2	3	4	5	6	7
0	:		0	1	2	3	4	5	6
1	:	7	8	9	A	B	C	D	E
2	:	-	F	G	H	I	J	K	L
3	:	M	N	O	P	Q	R	S	T
4	:	.	U	V	W	X	Y	Z	+
5	:	X	#	Γ	#	E	()	,
6	:	↑	%	!	/	=	*	X	̄
7	:	=	?	÷	!	II	Δ	II	Σ

Ainsi, le caractère « **A** » est obtenu avec le code **13**, le caractère « = » avec le code **64**...

Donc pour imprimer :

RAYON =

Il faut écrire :

OP 00	RAZ groupes
3 5	R (caractère 1)
1 3	A (caractère 2)
4 5	Y (caractère 3)
3 2	O (caractère 4)
3 1	N (caractère 5)
OP 01	affecte le groupe 1
0 0	Espace (caractère 6)
6 4	= (caractère 7)
0 0	Espace (caractère 8)
0 0	Espace (caractère 9)
0 0	Espace (caractère 10)
OP 02	affecte le groupe 2
OP 05	imprime la ligne

Nous pouvons aussi imprimer un texte de 5 caractères juste derrière le nombre présent à l'affichage (registre **x**) en utilisant **OP 04** (groupe 4) et **OP 06**.

Pour imprimer :

12 cm²

Il faut écrire :

OP 00
1 5
3 0
7 0
0 0
OP 04
1
2
OP 06

Programme complet

000	76	LBL	037	42	STO	074	00	00	111	00	00
001	11	A	038	03	03	075	69	OP	112	00	00
002	42	STO	039	71	SBR	076	02	02	113	00	00
003	01	01	040	30	TAN	077	69	OP	114	00	00
004	32	X/T	041	43	RCL	078	05	05	115	00	00
005	01	01	042	03	03	079	92	RTN	116	00	00
006	32	X/T	043	71	SBR	080	76	LBL	117	00	00
007	22	INV	044	28	LOG	081	38	SIN	118	69	OP
008	77	GE	045	71	SBR	082	69	OP	119	03	03
009	96	WRI	046	23	LNx	083	00	00	120	69	OP
010	71	SBR	047	25	CLR	084	03	03	121	05	05
011	23	LNx	048	91	R/S	085	03	03	122	92	RTN
012	71	SBR	049	76	LBL	086	01	01	123	76	LBL
013	39	COS	050	39	COS	087	07	07	124	30	TAN
014	43	RCL	051	69	OP	088	03	03	125	69	OP
015	01	01	052	00	00	089	05	05	126	00	00
016	71	SBR	053	03	03	090	02	02	127	03	03
017	28	LOG	054	05	05	091	04	04	128	06	06
018	65	*	055	01	01	092	03	03	129	04	04
019	02	02	056	03	03	093	00	00	130	01	01
020	65	*	057	04	04	094	69	OP	131	03	03
021	89	PI	058	05	05	095	01	01	132	05	05
022	95	=	059	03	03	096	01	01	133	02	02
023	42	STO	060	02	02	097	07	07	134	01	01
024	02	02	061	03	03	098	03	03	135	01	01
025	71	SBR	062	01	01	099	07	07	136	03	03
026	38	SIN	063	69	OP	100	03	03	137	69	OP
027	43	RCL	064	01	01	101	05	05	138	01	01
028	02	02	065	00	00	102	01	01	139	01	01
029	71	SBR	066	00	00	103	07	07	140	05	05
030	28	LOG	067	06	06	104	00	00	141	01	01
031	43	RCL	068	04	04	105	00	00	142	07	07
032	01	01	069	00	00	106	69	OP	143	00	00
033	33	X2	070	00	00	107	02	02	144	00	00
034	65	*	071	00	00	108	06	06	145	06	06
035	89	PI	072	00	00	109	04	04	146	04	04
036	95	=	073	00	00	110	00	00	147	00	00

LRN Programmer en LMS

148 00 00	179 02 02	210 02 02	241 69 OP
149 69 OP	180 06 06	211 99 PRT	242 02 02
150 02 02	181 04 04	212 22 INV	243 00 00
151 69 OP	182 06 06	213 58 FIX	244 00 00
152 05 05	183 04 04	214 92 RTN	245 03 03
153 92 RTN	184 06 06	215 76 LBL	246 01 01
154 76 LBL	185 04 04	216 96 WRI	247 03 03
155 23 LNX	186 06 06	217 69 OP	248 02 02
156 06 06	187 04 04	218 00 00	249 03 03
157 04 04	188 06 06	219 00 00	250 00 00
158 06 06	189 04 04	220 00 00	251 01 01
159 04 04	190 69 OP	221 03 03	252 04 04
160 06 06	191 03 03	222 06 06	253 69 OP
161 04 04	192 06 06	223 01 01	254 03 03
162 06 06	193 04 04	224 03 03	255 03 03
163 04 04	194 06 06	225 02 02	256 05 05
164 06 06	195 04 04	226 04 04	257 01 01
165 04 04	196 06 06	227 03 03	258 07 07
166 69 OP	197 04 04	228 06 06	259 00 00
167 01 01	198 06 06	229 69 OP	260 00 00
168 06 06	199 04 04	230 01 01	261 07 07
169 04 04	200 06 06	231 02 02	262 03 03
170 06 06	201 04 04	232 04 04	263 00 00
171 04 04	202 69 OP	233 03 03	264 00 00
172 06 06	203 04 04	234 05 05	265 69 OP
173 04 04	204 69 OP	235 00 00	266 04 04
174 06 06	205 05 05	236 00 00	267 69 OP
175 04 04	206 92 RTN	237 04 04	268 05 05
176 06 06	207 76 LBL	238 01 01	269 25 CLR
177 04 04	208 28 LOG	239 03 03	270 35 1/X
178 69 OP	209 58 FIX	240 01 01	271 91 R/S

Utilisation du programme :

rayon **A**

Le résultat sort sur l'imprimante.

Exemple :

Saisie : 1 5 A

Résultat :

```

=====
RAYON =                15.00
PERIMETRE =            94.25
SURFACE =              706.86
=====

```

Dans ce programme, nous constatons d'abord que des fonctions peuvent être utilisées en tant qu'étiquettes (labels) :

LBL COS, LBL LNX, LBL WRI...

et que des étiquettes peuvent être « appelées » par **SBR** (**SBR**) avec un retour après l'appel grâce à **RTN** (**INV SBR**), **SBR** et **RTN** signifiant respectivement *Subroutine* et *Return*.

Autre constat, nous voyons que l'impression de texte alphanumérique est coûteuse en « pas » de programmes :

- « RAYON = » routine **COS**, pas 49 à 79 = 31 pas
- « PERIMETRE = » routine **SIN**, pas 80 à 122 = 43 pas
- « SURFACE = » routine **TAN**, pas 123 à 153 = 31 pas
- « =====... » routine **LNX**, pas 154 à 206 = 53 pas
- « SAISIR UN NOMBRE ! » routine **WRI** 215 à 271 = 57 pas

Soit un total de 215 pas pour un programme de 271 pas !

Le programme comporte un test de valeur permettant d'aller vers un traitement d'erreur (**LBL WRI**) si la valeur du rayon saisi est inférieure à 1.

```
| 004 32 X/T |
| 005 01 01  |
| 006 32 X/T |
| 007 22 INV  |
| 008 77 GE   |
| 009 96 WRI  |
```

qui aurait pu s'écrire :

```
| 004 32 X/T |
| 005 00 00  |
| 006 77 GE   |
| 007 96 WRI  |
| 008 32 X/T |
```

GE et **INV GE** permettent un branchement conditionnel selon une comparaison entre les registres **x** et **t**, **GE** signifiant *Greater or Equal*.

La solution 1 (6 pas) met le rayon dans **t** par échange de **x** et de **t**, met la valeur « 1 » dans **x**, rééchange **x** et **t** pour avoir « 1 » dans **t** et le rayon dans **x** puis teste si **x** est strictement inférieur (**INV GE**) à **t** :

« Le rayon est-il strictement inférieur à 1 ? »

La solution 2 (5 pas) met le rayon dans **t** par échange de **x** et de **t**, met la valeur « 0 » dans **x** puis teste si **x** est plus grand ou égal à **t** :

« Zéro est-il plus grand ou égal au rayon ? »

(un échange de **x** et de **t**, après le test, remet le rayon dans **x** pour la suite des calculs, **RCL 01** étant plus coûteux d'un pas)

Deux autres particularités sont à expliciter :

1. La routine **LOG** permet l'impression du contenu du registre **x** en le formatant à deux décimales.

	207	76	LBL	
	208	28	LOG	
	209	58	FIX	
	210	02	02	
	211	99	PRT	
	212	22	INV	
	213	58	FIX	
	214	92	RTN	

FIX 2 fixe l'affichage à deux décimales, **PRT** imprime le registre **x** et **INV FIX** annule le formatage.

2. La routine **WRI**, impression du message d'erreur, se termine par :

	269	25	CLR	
	270	35	1/X	
	271	91	R/S	

CLR 1/X met le registre **x** à zéro et divise 1 par **x** ce qui provoque une erreur (division par zéro !) et déclenche le

clignotement de l'affichage pour signaler l'erreur, **R/S** arrêtant le programme.

(Cette « astuce » est souvent utilisée pour alerter l'utilisateur d'une erreur de saisie.)

Au regard des remarques précédentes, nous pouvons envisager de modifier ce programme pour l'améliorer, en effet les calculatrices concernées (**TI58**, **TI58C** et **TI59**) ayant une mémoire « programme » limitée en nombre de pas, un des principaux soucis de programmation est l'économie de pas, une démarche « d'économie » à outrance pouvant nuire à la lisibilité, donc à la maintenabilité, d'un programme...

Voici donc une version « optimisée » de ce programme :

000 69 OP	037 07 07	074 01 01	111 00 00
001 00 00	038 00 00	075 69 OP	112 69 OP
002 03 03	039 00 00	076 01 01	113 01 01
003 06 06	040 07 07	077 06 06	114 01 01
004 01 01	041 03 03	078 04 04	115 07 07
005 03 03	042 00 00	079 00 00	116 03 03
006 02 02	043 00 00	080 00 00	117 07 07
007 04 04	044 69 OP	081 00 00	118 03 03
008 03 03	045 04 04	082 00 00	119 05 05
009 06 06	046 69 OP	083 00 00	120 01 01
010 69 OP	047 05 05	084 00 00	121 07 07
011 01 01	048 25 CLR	085 69 OP	122 00 00
012 02 02	049 35 1/X	086 02 02	123 00 00
013 04 04	050 91 R/S	087 69 OP	124 69 OP
014 03 03	051 76 LBL	088 05 05	125 02 02
015 05 05	052 11 A	089 43 RCL	126 06 06
016 00 00	053 42 STO	090 01 01	127 04 04
017 00 00	054 01 01	091 71 SBR	128 65 *
018 04 04	055 32 X/T	092 28 LOG	129 06 06
019 01 01	056 00 00	093 65 *	130 22 INV
020 03 03	057 77 GE	094 02 02	131 28 LOG
021 01 01	058 00 00	095 65 *	132 95 =
022 69 OP	059 00 00	096 89 PI	133 69 OP
023 02 02	060 32 X/T	097 95 =	134 03 03
024 03 03	061 71 SBR	098 42 STO	135 69 OP
025 01 01	062 23 LNX	099 02 02	136 05 05
026 03 03	063 69 OP	100 69 OP	137 43 RCL
027 02 02	064 00 00	101 00 00	138 02 02
028 03 03	065 03 03	102 03 03	139 71 SBR
029 00 00	066 05 05	103 03 03	140 28 LOG
030 01 01	067 01 01	104 01 01	141 43 RCL
031 04 04	068 03 03	105 07 07	142 01 01
032 69 OP	069 04 04	106 03 03	143 33 X2
033 03 03	070 05 05	107 05 05	144 65 *
034 03 03	071 03 03	108 02 02	145 89 PI
035 05 05	072 02 02	109 04 04	146 95 =
036 01 01	073 03 03	110 03 03	147 42 STO

LRN Programmer en LMS

148 03 03	165 01 01	182 23 LNX	199 69 OP
149 69 OP	166 07 07	183 25 CLR	200 02 02
150 00 00	167 00 00	184 91 R/S	201 69 OP
151 03 03	168 00 00	185 76 LBL	202 03 03
152 06 06	169 06 06	186 23 LNX	203 69 OP
153 04 04	170 04 04	187 06 06	204 04 04
154 01 01	171 00 00	188 04 04	205 69 OP
155 03 03	172 00 00	189 06 06	206 05 05
156 05 05	173 69 OP	190 04 04	207 92 RTN
157 02 02	174 02 02	191 06 06	208 76 LBL
158 01 01	175 69 OP	192 04 04	209 28 LOG
159 01 01	176 05 05	193 06 06	210 58 FIX
160 03 03	177 43 RCL	194 04 04	211 02 02
161 69 OP	178 03 03	195 06 06	212 99 PRT
162 01 01	179 71 SBR	196 04 04	213 22 INV
163 01 01	180 28 LOG	197 69 OP	214 58 FIX
164 05 05	181 71 SBR	198 01 01	215 92 RTN

216 pas de programme au lieu de 272, soit une économie de 56 pas !

Le langage

Nous pouvons maintenant aborder les « verbes » par thèmes pour en faire le tour le plus exhaustif possible :

- Programmation
- Touches complémentaires
- Entrée des données
- Les opérations arithmétiques
- Effacement
- Racines et puissances
- Fonctions mathématiques
- Trigonométrie
- Impression
- Options d'affichage
- Gestion des données
- Branchements
- Statistiques
- Touches de fonctions
- Lecture / écriture
- Modules de librairie
- Opérations spéciales
- Autres fonctions
- L'instruction cachée

Programmation

- **CP** (**2nd** **CP**) en mode « calculatrice », efface toute la mémoire programme (mise à zéro de tous les pas), remet à zéro les adresses de retour des sous-programmes, ramène le pointeur de pas au pas 000 et efface le registre **t**.
- **LRN** (**LRN**) permet d'entrer en mode « programmation » ou d'en sortir (retour au mode « calculatrice »).
- **SST** (**SST**) en mode « programmation », avance d'un pas.
- **BST** (**BST**) en mode « programmation », recule d'un pas.
- **INS** (**2nd** **Ins**) en mode « programmation », insère un pas avant le pas en cours.
- **DEL** (**2nd** **Del**) en mode « programmation », supprime le pas en cours.

En mode programmation, l'appui sur une touche remplace l'instruction du pas en cours.

Touches complémentaires

- **2nd** (**2nd**) permet d'utiliser la deuxième fonction d'une touche correspondant à l'instruction mentionnée au dessus de la touche.

Exemple : **2nd** **SBR** donne **LBL**

- **INV** (**INV**) pour certaines fonctions (**EE**, **ENG**, **FIX**, **LOG**, **LNX**, **Yx**, **INT**, **SIN**, **COS**, **TAN**, **PRD**, **SUM**, **DMS**, **P/R**, **STA**, **AVR**, **LST**, **SBR**, **EQ**, **GE**, **IFF**, **STF**, **DSZ**, **WRI**), active la fonction inverse.

Dans certains cas, les deux touches **2nd** et **INV** pourront être utilisées.

Exemple : le logarithme décimal s'obtient en faisant **2nd** **Inx** et l'antilog du logarithme décimal s'obtient en faisant **INV** **2nd** **Inx**, que nous écrirons respectivement **LOG** et **INV LOG**.

Le mode « calculatrice » permet de taper aussi bien **2nd** **INV** **Inx** que **INV** **2nd** **Inx**, le mode programmation n'admettant que cette dernière notation (**INV** avant **2nd**) nous déconseillons de prendre l'habitude de la saisie inverse (**2nd** avant **INV**).

- **IND** (**2nd** **Ind**) permet l'adressage indirect des instructions de gestion des registres, des instructions de branchement et quelques autres l'instructions spécifiques.

Sont concernées par cette utilisation :

- Les instructions de gestion des registres **STO, RCL, EXC, SUM, INV SUM, PRD, INV PRD,**
- Les instructions de branchement **GTO, SBR, EQ, INV EQ, GE, INV GE, DSZ, INV DSZ, IFF, INV IFF**
- Les instructions spécifiques **PGM, OP, FIX, STF.**

L'adressage indirect permet d'utiliser le contenu d'un registre comme conteneur de l'adresse à utiliser.

Exemple :

5 STO 0 1 met la valeur 5 dans le registre 01,

5 STO 2nd Ind 0 1 met la valeur 5 dans le registre dont l'adresse est dans le registre 01. (Si le registre 01 contient 4, la valeur 5 sera stockée dans le registre 04)

Les instructions **DSZ** et **IFF** peuvent utiliser un double adressage indirect car elles manipulent à la fois un registre et une adresse de branchement.

INV 2nd 1 2nd Ind 0 1 2nd Ind 0 2

INV IFF IND 01 IND 02 signifie que si le drapeau, dont le numéro est contenu dans le registre 01, est baissé (flag=0) le programme ira à l'adresse qui est précisée dans le registre 02.








L'écriture des instructions avec adressage indirect pourra être différente de « instruction » suivi de **IND** selon le tableau suivant :

Séquence de touches	Instructions	Codes
STO 2nd Ind	ST*	72
RCL 2nd Ind	RC*	73
2nd RCL 2nd Ind	EX*	63
SUM 2nd Ind	SM*	74
INV SUM 2nd Ind	INV SM*	22 74
2nd SUM 2nd Ind	PR*	64
INV 2nd SUM 2nd Ind	INV PR*	22 64
GTO 2nd Ind	GT*	83
SBR 2nd Ind	SBR IND	71 40
2nd 7 2nd Ind	EQ	67 40
INV 2nd 7 2nd Ind	INV EQ	22 67 40
2nd 4 2nd Ind	GE	77 40
INV 2nd 4 2nd Ind	INV GE	22 70 40
2nd 0 2nd Ind	DSZ	97 40
INV 2nd 0 2nd Ind	INV DSZ	22 97 40
2nd 1 2nd Ind	IFF	87 40
INV 2nd 1 2nd Ind	INV IFF	22 87 40
2nd LRN 2nd Ind	PG*	62
2nd 9 2nd Ind	OP*	84
2nd (2nd Ind	FIX	58 40
2nd RST 2nd Ind	STF	86 40
INV 2nd RST 2nd Ind	INV STF	22 86 40

Entrée des données

- **Chiffres** (**0** **1** **2** ... **9**) introduction de chiffres dans le registre d'affichage **x**.
- **Point** (**.**) introduction du point décimal.
- **Signe** (**+/-**) change le signe du registre d'affichage **x**.
- **PI** (**2nd** **π**) introduit 3.14159265359 dans le registre d'affichage **x**.
- **|X|** (**2nd** **|x|**) retourne la valeur absolu du registre d'affichage **x**.
- **OP 10** (**2nd** **Op** **1** **0**) indique le sens de la valeur du registre d'affichage **x**
 Retourne **1** si $x > 0$, **0** si $x = 0$, **-1** si $x < 0$
- **INT** (**2nd** **Int**) retourne la partie entière du registre **x**.
- **INV INT** (**INV** **2nd** **Int**) retourne la partie décimale du registre **x**.

Les opérations arithmétiques

- / () division.
- * () multiplication.
- - () soustraction.
- + () addition.
- = () affiche et « fige » le résultat.
- (() parenthèse ouvrante.
-) () parenthèse fermante.

Les calculatrices **TI58/TI58C/TI59** utilisent la notation algébrique directe (système AOS).

Les opérations suivent donc la règle de priorité des opérateurs.

$2 + 3 * 4 =$ donnera comme résultat 14 tout comme $2 + (3 * 4) =$, les parenthèses étant, dans ce cas, inutiles.

Par contre $(2 + 3) * 4 =$ donnera comme résultat 20.

Plusieurs niveaux de parenthèses pourront être utilisés :




















$$2 + 3 * 4 / 5 = \quad \text{donnera } 2.8$$

$$((2 + 3) * 4) / 5 = \quad \text{donnera } 4$$

Effacement

- **CE** (**CE**) effacement de l'introduction en cours sans interférer sur les opérations en cours et arrête le clignotement de l'affichage.
- **CLR** (**CLR**) efface le registre **x** et les calculs en cours. Arrête aussi le clignotement de l'affichage.
- **CMS** (**2nd** **CMS**) efface tous les registres de donnée selon la partition définie (Voir **OP 16** et **OP 17**)
- **CP** (**2nd** **CP**) en mode programmation, efface uniquement le registre **t**.

Racines et puissances

- **X²** () élève au carré le contenu du registre d'affichage **x**.
- **SQR** () retourne la racine carrée du registre d'affichage **x**.
(si le registre **x** contient une valeur négative  provoque le clignotement de l'affichage)
- **Yx** () élève le nombre contenu dans le registre d'affichage à la puissance saisie :     donnera 1953125
- **INV Yx** () calcule la racine $x^{\text{ième}}$ du nombre contenu dans le registre d'affichage :
          donnera 5

Fonctions mathématiques

- **1/X** (**1/x**) calcule l'inverse du contenu du registre d'affichage **x**.
- **LNx** (**lnx**) calcule le logarithme népérien (base e) du registre d'affichage **x**. (si **x**<0 provoque le clignotement de l'affichage)
- **INV LNx** (**INV lnx**) calcule l'exponentielle (e^x) du registre d'affichage **x**.
- **LOG** (**2nd log**) calcule le logarithme décimal (base 10) du registre d'affichage **x**. (si **x**<0 provoque le clignotement de l'affichage)
- **INV LOG** (**INV 2nd log**) calcul l'antilog du registre d'affichage **x**. (élévation de 10 à la puissance **x**)
Souvent utilisé dans les programmes pour multiplier par un multiple de 10 plus grand que 100 :

$$\text{RCL 01} * \mathbf{1\ 0\ 0\ 0\ 0\ 0} = \text{coûte 10 pas}$$

$$\text{RCL 01} * \mathbf{5\ INV\ LOG} = \text{coûte 7 pas !}$$
- **P/R** (**2nd P→R**) convertit les coordonnées polaires en coordonnées cartésiennes à partir des registres **x** (*angle*) et **t** (*rayon*) et retourne l'ordonnée (*y*) dans le registre **x** et l'abscisse (*x*) dans le registre **t**.

Exemple :

10 x/t met le rayon dans le registre **t**
35 P/R met l'angle dans le registre **x**
 et retourne l'ordonnée 5.73576436351
x/t retourne l'abscisse 8.19152044289

- **INV P/R** (**INV** **2nd** **P→R**) convertit les coordonnées cartésiennes en coordonnées polaires à partir de l'ordonnée (y) dans le registre **x** et l'abscisse (x) dans le registre **t**, retourne l'angle dans le registre **x** et le rayon dans le registre **t**.

Il faudra veiller au choix du mode angulaire **DEG**, **RAD** ou **GRD** avant de procéder au calcul.

Le mode angulaire définit les limites de l'angle :

Mode angulaire	Borne inférieure	Borne supérieure
DEG	-90°	270°
RAD	$-\pi/2$	$3\pi/2$
GRD	-100	300

Trigonométrie

- **DEG** (**2nd** **Deg**) sélection mode angulaire « degré ».
- **RAD** (**2nd** **Rad**) sélection mode angulaire « radian ».
- **GRD** (**2nd** **Grad**) sélection mode angulaire « grade ».
- **SIN** (**2nd** **sin**) sinus du contenu du registre d'affichage **x**.
- **INV SIN** (**INV** **2nd** **sin**) Arc sinus du contenu du registre d'affichage **x**.
- **COS** (**2nd** **cos**) cosinus du contenu du registre d'affichage **x**.
- **INV COS** (**INV** **2nd** **cos**) Arc cosinus du contenu du registre d'affichage **x**.
- **TAN** (**2nd** **tan**) tangente du contenu du registre d'affichage **x**.
- **INV TAN** (**INV** **2nd** **tan**) Arc tangente du contenu du registre d'affichage **x**.

$$\text{Arc cosécante} = \mathbf{1/X INV SIN}$$

$$\text{Arc sécante} = \mathbf{1/X INV COS}$$

$$\text{Arc cotangente} = \mathbf{1/X INV TAN}$$

- **DMS** (**2nd** **D.Ms**) convertit un angle mesuré en degrés, minutes, secondes en degrés décimaux.

Le format de saisi est **DD.MMSSsss**, le point décimal doit séparer les degrés des minutes.

- **INV DMS** (**INV** **2nd** **D.Ms**) convertit un angle mesuré en degrés décimaux en degrés, minutes, secondes.

Impression

- **ADV** (**2nd** **Adv**) avance le papier d'une ligne.
- **PRT** (**2nd** **Prt**) imprime le registre **x**.
- **LST** (**2nd** **List**) liste le programme
- **INV LST** (**INV** **2nd** **List**) imprime le contenu des registres depuis le registre **nn** jusqu'au dernier, **nn** étant la valeur du registre **x**.
- **OP 00** (**2nd** **Op** **0** **0**) efface le tampon d'impression alphanumérique.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
OP 01					OP 02					OP 03					OP 04				

- **OP 01** (**2nd** **Op** **0** **1**) affecte le groupe 1 (extérieur gauche) du tampon d'impression alphanumérique.
- **OP 02** (**2nd** **Op** **0** **2**) affecte le groupe 2 (intérieur gauche) du tampon d'impression alphanumérique.
- **OP 03** (**2nd** **Op** **0** **3**) affecte le groupe 3 (intérieur droit) du tampon d'impression alphanumérique.

- **OP 04** (**2nd** **Op** **0** **4**) affecte le groupe 4 (extérieur droit) du tampon d'impression alphanumérique.
- **OP 05** (**2nd** **Op** **0** **5**) imprime le tampon d'impression.
- **OP 06** (**2nd** **Op** **0** **6**) imprime sur une même ligne le contenu du registre d'affichage **x** et les 4 derniers caractères du groupe 4 (extérieur droite) du tampon d'impression alphanumérique.

Le codage du tampon d'impression se fait à partir de la table suivante :

	0	1	2	3	4	5	6	7
0 :	0	1	2	3	4	5	6	7
1 :	7	8	9	A	B	C	D	E
2 :	-	F	G	H	I	J	K	L
3 :	M	N	O	P	Q	R	S	T
4 :	.	U	V	W	X	Y	Z	+
5 :	X	#	Γ	#	E	()	;
6 :	↑	%	!	/	=	*	X	Σ
7 :	=	?	÷	!	II	Δ	II	Σ

- **OP 07** (**2nd** **Op** **0** **7**) permet de tracer une courbe en imprimant une astérisques dans une colonne 0 à 19. Une seule astérisque est imprimé sur chaque ligne dans la colonne correspondant à la partie entière du registre d'affichage **x** dans l'intervalle de valeur $-1 < \mathbf{x} < 20$.

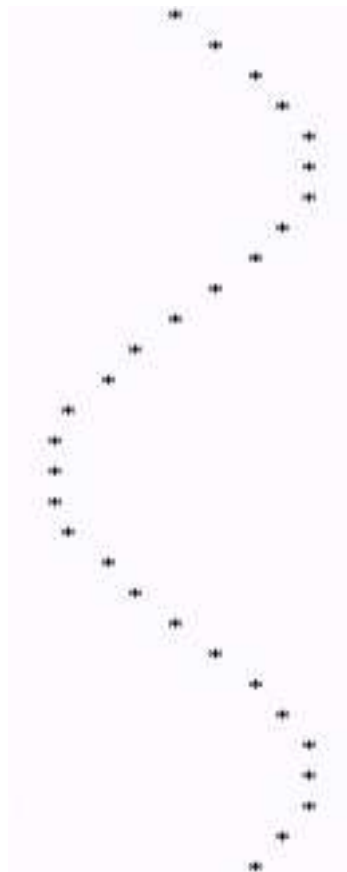
Exemple :

Sinusoïde de 18 degrés en 18 degrés.

```

000 76 LBL
001 11 A
002 43 RCL
003 01 01
004 38 SIN
005 85 +
006 01 01
007 95 =
008 65 *
009 09 09
010 93 .
011 09 09
012 95 =
013 69 OP
014 07 07
015 01 01
016 08 08
017 44 SUM
018 01 01
019 11 A

```



+1 = * 9.9 = permet de recaler la valeur dans un intervalle de 0 à 19.8 pour déterminer la colonne de l'astérisque.

- **OP 08** (**2nd** **Op** **0** **8**) liste des étiquettes (labels) du programme.

```
001 25 CLR
015 15 E
108 11 A
191 12 B
267 35 1/X
294 23 LNX
322 24 CE
354 45 YX
423 33 X2
```

Options d'affichage

L'affichage standard des TI se fait sur 10 chiffres alors que la gestion interne est sur 13 chiffres pour plus de précision dans les calculs.

L'affichage est donc limité aux nombres compris entre .0000000001 et 9999999999 (en valeur absolu, le signe ne prenant pas de position sur les 10 caractères).

Les nombres dépassant ces limites devront être saisis en notation scientifique.

Ainsi le nombre

-0.0000000000000000000000000000001234567

peut s'écrire

-1.234567 * 10⁻³¹

et sera introduit de la façon suivante

1.234567 +/- **EE** **31** +/-

et s'affichera



-1.234567 étant la *mantisse* et **-31** étant l'*exposant*

La mantisse est donc limitée à 7 caractères et l'exposant à 2.

- **EE** (**EE**) permet de passer en notation scientifique
- **INV EE** (**INV EE**) permet l'annulation de la notation scientifique.
- **ENG** (**2nd Eng**) permet de passer en notation « ingénieur ». Variante de la notation scientifique, la notation « ingénieur » se caractérise par un ajustement de la mantisse et de l'exposant afin d'avoir un exposant multiple de trois. Ainsi $-1.234567-31$ donnera $-123.4567-33$ en notation « ingénieur ».
- **INV ENG** (**INV 2nd Eng**) permet l'annulation de la notation « ingénieur ».

La notation « ingénieur » permet de représenter les nombres dans des unités de mesure usuelles :

10ⁿ	Préfixe	Nombre décimal
10 ²⁴	Yotta	1 000 000 000 000 000 000 000 000
10 ²¹	Zetta	1 000 000 000 000 000 000 000
10 ¹⁸	Exa	1 000 000 000 000 000 000
10 ¹⁵	Péta	1 000 000 000 000 000
10 ¹²	Téra	1 000 000 000 000
10 ⁹	Giga	1 000 000 000
10 ⁶	Méga	1 000 000
10 ³	Kilo	1 000
10 ²	Hecto	100
10 ¹	Déca	10
10 ⁰	<i>Unité</i>	1
10 ⁻¹	Déci	0,1
10 ⁻²	Centi	0,01
10 ⁻³	Milli	0,001
10 ⁻⁶	Micro	0,000 001
10 ⁻⁹	Nano	0,000 000 001
10 ⁻¹²	pico	0,000 000 000 001
10 ⁻¹⁵	femto	0,000 000 000 000 001
10 ⁻¹⁸	atto	0,000 000 000 000 000 001
10 ⁻²¹	zepto	0,000 000 000 000 000 000 001
10 ⁻²⁴	yocto	0,000 000 000 000 000 000 000 001

- **FIX** (**2nd** **Fix**) permet de choisir la décimalisation.

Le chiffre suivant la touche **FIX** indique le nombre de décimales fixes (0 à 8).

- **FIX IND** (**2nd** **Fix** **2nd** **Ind**) permet de choisir, ou annuler, la décimalisation de façon indirecte.

Le chiffre suivant la touche **FIX** indique le numéro du registre contenant le nombre de décimales fixes (0 à 8), ou la valeur 9 pour repasser en virgule flottante.

- **INV FIX** (**INV** **2nd** **Fix**) annule la décimalisation et repasse en virgule flottante. (**FIX 9** a le même effet)

Gestion des données

- **X/T** (**x=t**) échange les contenus des registres **x** et **t**.
- **STO** (**STO**) stocke le contenu du registre **x** dans un registre nn.
- **ST*** (**STO** **2nd** **Ind**) stocke le contenu du registre **x** dans un registre dont l'adresse est contenue dans le registre nn.

1 5 **STO** **2nd** **Ind** 0 1

1 5 ST* 01 met la valeur 15 dans le registre dont l'adresse est stockée dans le registre 01.

Si le registre 01 contient 20, met 15 dans le registre 20,

Si le registre 01 contient 7 met 15 dans le registre 7...

- **RCL** (**RCL**) met le contenu du registre nn dans le registre **x**.
- **RC*** (**RCL** **2nd** **Ind**) met le contenu du registre dont l'adresse est contenue dans le registre nn dans le registre **x**.
- **SUM** (**SUM**) additionne le contenu du registre **x** au contenu du registre nn.
- **SM*** (**SUM** **2nd** **Ind**) additionne le contenu du registre **x** au contenu du registre dont l'adresse est contenue dans le registre nn.

- **INV SUM** (**INV** **SUM**) soustrait le contenu du registre **x** du contenu du registre **nn**.
- **INV SM*** (**INV** **SUM** **2nd** **Ind**) soustrait le contenu du registre **x** du contenu du registre dont l'adresse est contenue dans le registre **nn**.
- **PRD** (**2nd** **Prd**) multiplie le contenu du registre **nn** par le contenu du registre **x**.
- **PD*** (**2nd** **Prd** **2nd** **Ind**) multiplie le contenu du registre dont l'adresse est contenue dans le registre **nn** par le contenu du registre **x**.
- **INV PRD** (**INV** **2nd** **Prd**) divise le contenu du registre **nn** par le contenu du registre **x**.
- **INV PD*** (**INV** **2nd** **Prd** **2nd** **Ind**) divise le contenu du registre dont l'adresse est contenue dans le registre **nn** par le contenu du registre **x**.
- **EXC** (**2nd** **Exc**) échange le contenu du registre **nn** avec le contenu du registre **x**.

- **EX*** (**2nd** **Exc** **2nd** **Ind**) échange le contenu du registre dont l'adresse est contenue dans le registre nn avec le contenu du registre x.

- **OP 2n** (**2nd** **Op** **2** **n**) incrémente la mémoire du registre n de 1. S'applique aux registres 0 à 9.

OP 21 est équivalent à **1 SUM 01**

- **OP 3n** (**2nd** **Op** **3** **n**) décrémente la mémoire du registre n de 1. S'applique aux registres 0 à 9.

OP 31 est équivalent à **1 INV SUM 01**

Branchements

- **LBL** (**2nd** **Lbl**) permet de définir des étiquettes de programme (ou labels).

2 types d'étiquettes sont utilisables :

- les étiquettes « utilisateur » (ou touches de fonctions) : **A**, **B**, **C**...
- les étiquettes ordinaires : toutes les touches peuvent alors être utilisées comme labels à l'exception des touches numériques (**0**, **1**, **2**...) et des touches **2nd**, **LRN**, **BST**, **SST**, **2nd** **Ins**, **2nd** **Del**, **2nd** **Ind** et de la touche **R/S** (autorisée mais fortement déconseillée).

Bien entendu, dans le cas d'utilisation d'une touche en tant que label, cette dernière ne sera pas traitée comme instruction dans le déroulement du programme mais uniquement comme label.

- **GTO** (**GTO**) permet le branchement à une adresse précise. Déplace le pointeur de pas à l'adresse indiquée et, en mode programmation, continue l'exécution du programme à partir de cette adresse.

Deux adressage sont possibles

- Adressage logique : **GTO** est alors suivi d'un nom défini par ailleurs comme label.

Exemple :

GTO **x²**... et ailleurs dans le programme **2nd** **Lbl** **x²**...

- Adressage absolu : **GTO** est alors suivi d'un numéro de pas.

Exemple :

GTO **1** **2** **3** qui renvoie au pas 123

L'avantage de l'adressage logique réside dans la clarté et la lisibilité du programme, et l'ajout, ou la suppression, de pas dans le programme ne change rien au lien logique. (Cette méthode coûte au minimum 4 pas.)

L'adressage absolu permet une économie de pas (3 pas) mais impose une vigilance quant à la maintenance car des ajouts, ou des suppressions, de pas dans le programme décale d'autant l'adresse visée par le **GTO** si ces mises à jour interviennent avant cette adresse.

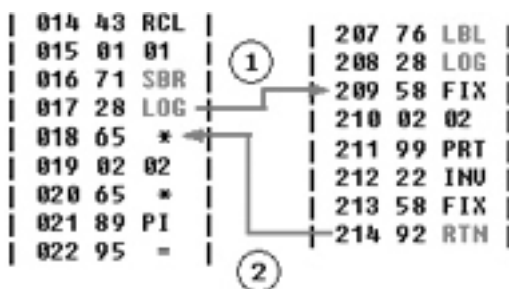
- **GO*** (**GTO** **2nd** **Ind**) permet l'adressage relatif dans un programme en utilisant un registre de donnée qui contient l'adresse du pas visé par le branchement.

Exemple :

GTO **2nd** **Ind** **0** **1** signifie que l'adresse de branchement est contenue dans le registre **01**.

- **SBR** (**SBR**) permet le branchement à l'adresse précisée comme pour **GTO** mais la première instruction de retour **RTN** (**INV** **SBR**) renverra le pointeur derrière l'appel **SBR**.

SBR utilise, comme **GTO**, l'adressage logique et l'adressage absolu.

Exemple :

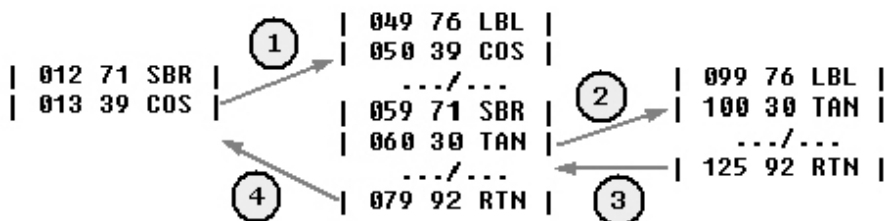
- ① appel de la procédure commençant à l'étiquette **LOG**,
- ② retour derrière l'appel.

- **SBR IND** (**SBR** **2nd** **Ind**) permet l'appel relatif dans un programme en utilisant un registre de donnée qui contient l'adresse du pas visé par l'appel de procédure.

La première instruction de retour **RTN** (**INV** **SBR**) renverra le pointeur derrière l'appel **SBR**.

- **RTN** (**INV** **SBR**) retour de procédure appelée par **SBR** (*Return*).

Dans le cas où l'exécution rencontre une instruction **RTN** alors qu'aucune instruction **SBR** est en attente de retour, alors **RTN** se comporte comme **R/S** et arrête le programme.



Dans le cas d'appels imbriqués, le retour se fait derrière le dernier appel effectué et ainsi de suite jusqu'à l'épuisement de la pile d'adresses de retour.

- **RST** (**RST**) ramène le pointeur de pas au pas 000, remet à zéro les adresses de retour des sous-programmes et remet les drapeaux (*flags*) à zéro (« position basse »).

- **R/S** (**R/S**) en mode « calculatrice » lance le programme à partir du pointeur courant ou arrête le programme, en mode programme arrête le programme.

- **EQ** (**2nd** **x=t**) test conditionnel, va à l'adresse précisée si le registre **x** est égal au registre **t**, sinon le programme continue en séquence.

EQ utilise, comme **GTO**, l'adressage logique et l'adressage absolu.

Exemple :

2nd **x=t** **lnx**

va à l'étiquette **LNx** si **x = t**

2nd **x=t** **1** **2** **3**

va à l'adresse **123** si **x = t**

- **EQ IND** (**2nd** **x=t** **2nd** **Ind**) test conditionnel, utilisant un registre de donnée qui contient l'adresse du pas visé si le registre **x** est égal au registre **t**, sinon le programme continue en séquence.
- **INV EQ** (**INV** **2nd** **x=t**) test conditionnel, va à l'adresse précisée si le registre **x** est différent du registre **t**, sinon le programme continue en séquence.
- **INV EQ IND** (**INV** **2nd** **x=t** **2nd** **Ind**) test conditionnel, utilisant un registre de donnée qui contient l'adresse du pas visé si le registre **x** est différent du registre **t**, sinon le programme continue en séquence.
- **GE** (**2nd** **x ≥ t**) test conditionnel, va à l'adresse précisée si le registre **x** est supérieur ou égal au registre **t**, sinon le programme continue en séquence.
GE utilise, comme **GTO**, l'adressage logique et l'adressage absolu.
- **GE IND** (**2nd** **x ≥ t** **2nd** **Ind**) test conditionnel, utilisant un registre de donnée qui contient l'adresse du pas visé si le registre **x** est supérieur ou égal au registre **t**, sinon le programme continue en séquence.

- **INV GE** (**INV** **2nd** **$x \geq t$**) test conditionnel, va à l'adresse précisée si le registre **x** est inférieur au registre **t**, sinon le programme continue en séquence.
- **INV GE IND** (**INV** **2nd** **$x \geq t$** **2nd** **Ind**) test conditionnel, utilisant un registre de donnée qui contient l'adresse du pas visé si le registre **x** est inférieur au registre **t**, sinon le programme continue en séquence.

Branchements conditionnels			
Egal	EQ	2nd	$x=t$
Différent	INV EQ	INV 2nd	$x=t$
Supérieur ou égal	GE	2nd	$x \geq t$
Inférieur	INV GE	INV 2nd	$x \geq t$

Hormis les tests conditionnels par comparaison des registres **x** et **t**, la TI permet de gérer jusqu'à 10 drapeaux (*flags*) dont l'état (levé ou baissé) pourra être testé pour branchement.

Les drapeaux sont numérotés de 0 à 9.

- **STF** (**2nd** **St flg**) lève le drapeaux précisé (*Set Flag*).

Exemple :

2nd **St flg** **1** lève le drapeau 1

- **INV STF** (**INV** **2nd** **St flg**) baisse le drapeau précisé.

Exemple :

INV **2nd** **St flg** **1** baisse le drapeau 1

- **IFF** (**2nd** **If flg**) test conditionnel, va à l'adresse précisée si le drapeau précisé est levé, sinon le programme continue en séquence.

IFF utilise, comme **GTO**, l'adressage logique et l'adressage absolu.

- **IFF IND** (**2nd** **If flg** **2nd** **Ind**) test conditionnel, utilisant un registre de donnée qui contient l'adresse du pas visé si le drapeau précisé est levé, sinon le programme continue en séquence.

- **INV IFF** (**INV** **2nd** **If flg**) test conditionnel, va à l'adresse précisée si le drapeau précisé est baissé, sinon le programme continue en séquence.

- **INV IFF IND** (**INV** **2nd** **If fig** **2nd** **Ind**) test conditionnel, utilisant un registre de donnée qui contient l'adresse du pas visé si le drapeau précisé est baissé, sinon le programme continue en séquence.
- **DSZ** (**2nd** **Dsz**) test conditionnel permettant de gérer des boucles. manipule un registre de donnée (0 à 9 uniquement) et utilise, comme **GTO**, l'adressage logique et l'adressage absolu.

DSZ procède en deux étapes :

- Décrément du registre testé si valeur positive (ou incrément si valeur négative)
- Test si registre à zéro : si **NON** va à l'adresse précisée, si **OUI** continue en séquence.

• **DSZ IND** (**2nd** **Dsz** **2nd** **Ind**) test conditionnel permettant de gérer des boucles. manipule un registre de donnée (0 à 9 uniquement) et utilise un registre de donnée qui contient l'adresse du pas visé si le test est satisfait.

• **INV DSZ** (**INV** **2nd** **Dsz**) test conditionnel permettant de gérer des boucles. manipule un registre de donnée (0 à 9 uniquement) et utilise, comme **GTO**, l'adressage logique et l'adressage absolu.

INV DSZ procède en deux étapes :

- Décrément du registre testé si valeur positive (ou incrément si valeur négative)
 - Test si registre à zéro : si **OUI** va à l'adresse précisée, si **NON** continue en séquence.
- **INV DSZ IND** (**INV** 2nd **Dsz** 2nd **Ind**) test conditionnel permettant de gérer des boucles. manipule un registre de donnée (0 à 9 uniquement) et utilise un registre de donnée qui contient l'adresse du pas visé si le test est satisfait.

Statistiques

La TI gère les statistiques pour un échantillon sur deux valeurs représentant un point sur un plan d'axes x et y .

Sur la population de points, nous pouvons déterminer la moyenne, la variance, l'écart type...

- Initialisation des données statistiques : les statistiques utilisent 6 registres de données, et le registre t , qui devront être remis à zéro avant toute nouvelle introduction.

Registre	01	02	03	04	05	06
Contenu	Σy	Σy^2	N	Σx	Σx^2	Σxy

Cette initialisation peut être faite :

- Soit manuellement : **2nd** **CMs** qui efface tous les registres,
- Soit manuellement : **CLR** **STO** **0** **1** **STO** **0** **2** **STO** **0** **3** **STO** **0** **4** **STO** **0** **5** **STO** **0** **6**,
- Soit en utilisant la routine d'initialisation du module 01 de la bibliothèque de base (**ML-01**) : **2nd** **Pgm** **0** **1** **SBR** **CLR**

- **STA** (**2nd** **Σ+**) introduction des données.

Soit :

- **x** **↵** **y** **2nd** **Σ+** pour introduire **x** et **y**
- **y** **2nd** **Σ+** pour introduire **y** seul

le rang **i** est affiché pour chaque couple (**x_i**, **y_i**) introduit.

- **INV STA** (**INV** **2nd** **Σ+**) annulation de données.

Soit :

- **x** **↵** **y** **2nd** **Σ+** pour annuler **x** et **y**
- **y** **2nd** **Σ+** pour annuler **y** seul

- **AVR** (**2nd** **\bar{x}**) calcule et affiche la moyenne des différentes valeurs de **y** (**↵** pour afficher la moyenne des différentes valeurs de **x**).

- **INV AVR** (**INV** **2nd** **\bar{x}**) calcule et affiche l'écart type des différentes valeurs de **y** (**↵** pour afficher l'écart type des différentes valeurs de **x**).

- **OP 11** (**2nd** **Op** **1** **1**) calcule et affiche la variance des différentes valeurs de **y** (**↵** pour afficher la variance des différentes valeurs de **x**).

- **OP 12** (2nd Op 1 2) **Régression linéaire** – calcule et affiche l'ordonnée à l'origine (point d'intersection de la droite de régression avec l'axe des y pour $x = 0$) et $\frac{\Delta y}{\Delta x}$ affiche la pente.
- **OP 13** (2nd Op 1 3) **Régression linéaire** – calcule et affiche le coefficient de corrélation.
- **OP 14** (2nd Op 1 4) **Régression linéaire** – calcul et affiche la valeur de y pour une valeur de x saisi.
- **OP 15** (2nd Op 1 5) **Régression linéaire** - calcul et affiche la valeur de x pour une valeur de y saisi.

Touches de fonctions

Les touches de fonctions (ou touches utilisateur) sont au nombre de 10. Elles sont utilisables dans les programmes en tant que label et peuvent être appelés par les instructions de branchement (**GTO**, **GE**, **EQ...**).

L'utilisation d'une touche seule équivaut à **SBR**. (**Exemple** : **SBR**
A = **A**)

En mode « calculatrice », elles permettent le lancement à partir d'un point précis du programme.

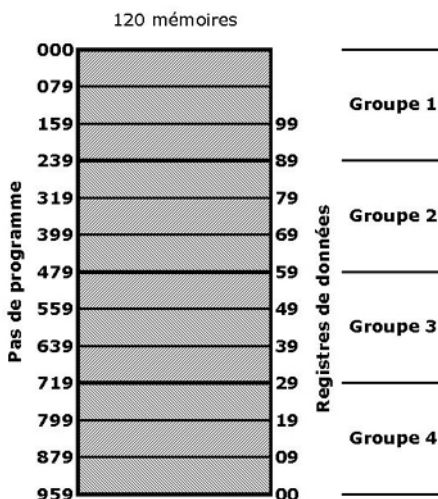
- **A** (**A**)
- **B** (**B**)
- **C** (**C**)
- **D** (**D**)
- **E** (**E**)

- **A'** (**2nd** **A'**)
- **B'** (**2nd** **B'**)
- **C'** (**2nd** **C'**)
- **D'** (**2nd** **D'**)
- **E'** (**2nd** **E'**)

Lecture / écriture

Les instructions de lecture / écriture sont utilisables uniquement sur **TI59** car elle est la seule à être dotée d'un lecteur de cartes magnétiques.

La **TI59** possède jusqu'à 120 mémoires de stockage réparties entre 4 groupes.



Une carte magnétique pour **TI59** contient 2 pistes pouvant enregistrer chacune 1 groupe.

Deux cartes sont donc nécessaires pour enregistrer toute la mémoire d'une **TI59**.

- **WRI** (**2nd** **Write**) écriture sur la carte (doit être précédé du numéro de groupe à enregistrer 1, 2, 3 ou 4)

- **INV WRI** (**INV** **2nd** **Write**) lecture de la carte (si précédé du numéro de groupe signé en négatif **-n**, force la lecture dans le groupe n)

Modules de librairie

Avec la calculatrice, un module enfichable est fourni en standard.

Nommé « Master Library », il contient vingt-cinq programmes utilitaires.

Il peut être remplacé par un autre des modules commercialisés par Texas Instruments.



No	Code	Intitulé
01	ML	Master Library
02	ST	Applied Statistics
03	RE	Real Estate / Investment
04	SY	Surveying
05	NG	Marine Navigation
06	AV	Aviation
07	LE	Leisure Library
08	SA	Securities Analysis
09	BD	Business Decisions
10	MU	Math / Utilities
11	EE	Electrical Engineering
12	FM	Agriculture
13	RP	RPN Simulator

- **PGM** (**2nd Pgm**) permet d'activer, ou désactiver, un programme de la bibliothèque.
 - **2nd Pgm nn** active le programme nn,
 - **2nd Pgm 0 0** désactive le programme en cours.

2nd Pgm 0 1 SBR 2nd Write permet d'afficher le numéro du module enfiché et imprime son nom si l'imprimante est connectée.

Exemple :

Le programme 24 de la « Master Library » convertit des mesures décimales (cm, m, km) en mesures anglo-saxonnes (inch, foot, yard, miles)

Ainsi pour savoir combien 1 yard fait de mètres il faut introduire la séquence :

2nd Pgm 2 4 1 C

ML-24 UNIT CONVERSIONS (I)				
cm → in	m → ft	m → yd	km → mi	n.mi → mi
in → cm	ft → m	yd → m	mi → km	mi → n.mi

- **OP 09** (**2nd Op 0 9**) charge le programme activé dans la mémoire programme de la TI. (efface le programme en mémoire pour le remplacer !)

Opérations spéciales

- **OP 01 à OP 08** voir *Impression*
- **OP 09** voir *Modules de librairie*
- **OP 10** voir *Entrée des données*
- **OP 11 à OP 15** voir *Statistiques*
- **OP 16** (2nd **Op** 1 6) affiche la partition mémoire : répartition entre pas de programme et registres de donnée.
- **OP 17** (2nd **Op** 1 7) positionne la partition mémoire : répartition entre pas de programme et registres de donnée par groupe de 10 registres.

T159	OP 17	T158/T158C
959-00	0	479-00
879-09	1	399-09
799-19	2	319-19
719-29	3	239-29
639-39	4	159-39
559-49	5	079-49
479-59	6	000-59
399-69	7	
319-79	8	
239-89	9	
159-99	10	

Exemple :

Sur **T158**, 3 2nd **Op** 1 7 donnera **239.29** qui signifie 240 pas (de 000 à 239) et 30 registres (de 00 à 29)

- **OP 18** (**2nd** **Op** **1** **8**) lève le drapeau 7 si aucune erreur d'exécution n'est rencontrée.
- **OP 19** (**2nd** **Op** **1** **9**) lève le drapeau 7 si une erreur d'exécution est rencontrée.
- **OP 40** (**2nd** **Op** **4** **0**) sur **TI58C** uniquement, lève le drapeau 7 si l'imprimante est connectée.
- **OP IND** (**2nd** **Op** **2nd** **Ind**) utilise le contenu d'un registre nn pour déterminer quelle opération est applicable.

Exemple :

2nd **Op** **2nd** **Ind** **0** **1** utilise le contenu du registre 01.

- Si le registre 01 contient 16, affiche la partition (idem **OP 16**),
- Si le registre 01 contient 0, efface les tampons d'impression (idem **OP 00**).

Autres fonctions

- **PAU** (**2nd Pause**) permet de conserver l'affichage du registre **x** pendant une demi seconde durant l'exécution du programme. Plusieurs pauses peuvent se succéder pour prolonger l'affichage.
- **NOP** (**2nd Nop**) pas d'opération. Instruction sans aucun effet sur l'exécution. Sert à insérer un pas pour prévoir un espace entre deux séquences de programme ou pour remplacer une instruction sans décalage dans la numérotation des pas au lieu de faire **DEL**.

L'instruction cachée

• **HIR** (*pas de touche*) Les **TI59/58/58C** cachent 8 registres internes utilisés par le système pour ses propres fonctions.

Le système basé sur la notation algébrique directe gère une pile AOS dans ces registres pour mettre en attente les nombres dans les calculs à plusieurs opérateurs afin de respecter la priorité de ces opérateurs.

Ensuite des fonctions complexes (**STA, AVR, P/R, DMS**) stockent des résultats intermédiaires dans ces registres ainsi que les fonctions statistiques (**OP 11, OP 12, OP 13, OP 14, OP 15**) et les fonctions d'impression alphanumérique (**OP 00, OP 01, OP 02, OP 03, OP 04**).

Il existe une instruction particulière pour manipuler ces registres. Officiellement, cette instruction n'existe pas :

- Aucune mention dans les documentations des **TI**,
- Aucune touche pour l'introduire dans un programme.

Et pourtant...

Il faut ainsi ruser pour introduire cette instruction avec des manipulations qui s'apparentent plus à du jonglage qu'à de la programmation.

Saisissons donc un petit programme...

Nous passons en mode programmation (**LRN**) après avoir effacé le contenu de la mémoire programme (**2nd CP**).

Introduisons les instructions suivantes :

2nd **Lbl** **A** **STO** **8** **2** **STO** **1** **1** **R/S**

Ce qui donne avec **2nd** **List** :

```

000 76 LBL
001 11 A
002 42 STO
003 82 82
004 42 STO
005 11 11
006 91 R/S

```

Modifions maintenant notre programme en supprimant le pas 004 puis le pas 002 :

- **RST**, **LRN** puis **SST** **SST** **SST** **SST** pour aller au pas 004
- **2nd** **Del** pour supprimer le pas 004
- **BST** **BST** pour aller au pas 002
- **2nd** **Del** pour supprimer le pas 002.

Nous obtenons :

```

000 76 LBL
001 11 A
002 82 HIR
003 11 11
004 91 R/S

```

Nous constatons que le code **82** a été traduit en **HIR** par l'imprimante.

Voici donc notre instruction cachée qui apparaît.

En mode « calculatrice », entrons le petit calcul suivant :

7 **+** **3** **×** **4** **=**

qui nous donne 19 car la multiplication est prioritaire sur l'addition.

Maintenant exécutons notre petit programme en tapant sur la touche de fonction **A**.

Le chiffre 7 apparaît à l'affichage.

Il s'agit du premier chiffre de notre calcul qui a été mis en attente (stocké dans la pile AOS) pour que la multiplication puisse être effectuée en premier.

HIR 12 ferait apparaître 3 à l'affichage, nous montrant ainsi que le deuxième chiffre de notre opération a aussi été stocké dans la pile AOS.

- **HIR 0n** ($0 \leq n \leq 8$) transfère le contenu du registre d'affichage **x** dans le registre interne **n**. (\approx **STO**)
- **HIR 1n** ($0 \leq n \leq 8$) transfère le contenu du registre interne **n** dans le registre d'affichage **x**. (\approx **RCL**)
- **HIR 3n** ($0 \leq n \leq 8$) additionne le contenu du registre d'affichage **x** au registre interne **n**. (\approx **SUM**)

- **HIR 4n** ($0 \leq n \leq 8$) multiplie le contenu du registre interne n par le contenu du registre d'affichage x . (\approx **PRD**)
- **HIR 5n** ($0 \leq n \leq 8$) soustrait le contenu du registre d'affichage x du registre interne n . (\approx **INV SUM**)
- **HIR 6n** ($0 \leq n \leq 8$) divise le contenu du registre interne n par le contenu du registre d'affichage x . (\approx **INV PRD**)

Tableau récapitulatif des instructions

Code	Instr.	Touches
00	0	0
01	1	1
02	2	2
03	3	3
04	4	4
05	5	5
06	6	6
07	7	7
08	8	8
09	9	9
10	E'	2nd E'
11	A	A
12	B	B
13	C	C
14	D	D
15	E	E
16	A'	2nd A'
17	B'	2nd B'
18	C'	2nd C'
19	D'	2nd D'
20	CLR	CLR
21	2nd	2nd
22	INV	INV
23	LNx	lnx
24	CE	CE
25		
26		
27		
28	LOG	2nd log
29	CP	2nd CP
30	TAN	2nd tan

Code	Instr.	Touches
31	LRN	LRN
32	X/T	x^{±t}
33	X2	x²
34	SQR	√x
35	1/X	1/x
36	PGM	2nd Pgm
37	P/R	2nd P→R
38	SIN	2nd sin
39	COS	2nd cos
40	IND	2nd Ind
41	SST	SST
42	STO	STO
43	RCL	RCL
44	SUM	SUM
45	YX	y^x
46	INS	2nd Ins
47	CMS	2nd CMS
48	EXC	2nd Exc
49	PRD	2nd Prd
50	IXI	2nd x
51	BST	BST
52	EE	EE
53	((
54))
55	/	÷
56	DEL	2nd Del
57	ENG	2nd Eng
58	FIX	2nd Fix
59	INT	2nd Int
60	DEG	2nd Deg
61	GTO	GTO

Code	Instr.	Touches
62	PG*	2nd Pgm 2nd Ind
63	EX*	2nd Exc 2nd Ind
64	PR*	2nd Prd 2nd Ind
65	*	X
66	PAU	2nd Pause
67	EQ	2nd x=t
68	NOP	2nd Nop
69	OP	2nd Op
70	RAD	2nd Rad
71	SBR	SBR
72	ST*	STO 2nd Ind
73	RC*	RCL 2nd Ind
74	SM*	SUM 2nd Ind
75	-	-
76	LBL	2nd Lbl
77	GE	2nd x ≥ t
78	STA	2nd Σ+
79	AVR	2nd \bar{x}
80	GRD	2nd Grad

Code	Instr.	Touches
81	RST	RST
82	HIR	
83	GO*	GTO 2nd Ind
84	OP*	2nd Op 2nd Ind
85	+	
86	STF	2nd St flg
87	IFF	2nd If flg
88	DMS	2nd D.Ms
89	PI	2nd Π
90	LST	2nd List
91	R/S	R/S
92	RTN	INV SBR
93	.	.
94	+/-	+/-
95	=	=
96	WRI	2nd Write
97	DSZ	2nd Dsz
98	ADV	2nd Adv
99	PRT	2nd Prt

Tests Comparatifs

Pour une même fonctionnalité, plusieurs solutions de programmation peuvent se présenter.

Le coût en nombre de pas ou le temps d'exécution peuvent influencer le choix de programmation selon le cas étudié.

Parfois, l'économie de pas pourra s'avérer cruciale, la mémoire étant relativement limitée.

Occasionnellement le temps d'exécution sera privilégié comme critère d'optimisation.

Heureusement, eu égard la nature même des programmes développés pour ce genre de machine, ces préoccupations seront le plus souvent superflues.

Néanmoins, l'étude des différentes hypothèses de résolution de cas de programmation reste utile à la compréhension des mécanismes du langage.

Remise à zéro des registres

Un grand classique de programmation avec ce type de machine est la remise à zéro de certains registres.

En effet, pour réinitialiser tous les registres d'un coup nous avons l'instruction **2nd CMs** qui répond à tout les critères possibles : rapidité et 1 seul pas de programme.

Mais pour remettre à zéro un ensemble de registres nous aurons trois choix de programmation :

- Programmation par décrémentation,
- Manipulation des partitions,
- Utilisation des programmes de bibliothèques.

Les 3 méthodes abordées sont sur la base d'une remise à zéro des registres 00 à 09 et des registres 00 à 29.

1^{ère} méthode : Programmation par décrémentation

```

...   n
...   n      nn = registre max (9 ou 29)
...   STO
...   00
...   CLR
xxx   ST*
...   00
...   DSZ
...   0
...   0x     xxx = adresse de branchement
...   xx

```

2^{ème} méthode : Manipulation des partitions

... **n**
 ... **OP** n = nombre de groupes de mémoire
 ... **17** (1 pour 00 à 09, 3 pour 00 à 29 [*])
 ... **CMS**
 ... **m**
 ... **OP** m = retour à la partition initiale
 ... **17** (5 par exemple pour 159-39 [*])

[*] concerne les **TI58** et **TI58C**

3^{ème} méthode : Utilisation des programmes de bibliothèques

... **n**
 ... **n** nn = registre max (9 ou 29)
 ... **PGM**
 ... **01**
 ... **SBR**
 ... **00**
 ... **12** Utilisation Module de base ML-01

ou

... **n**
 ... **n** nn = registre max (9 ou 29)
 ... **PGM**
 ... **01**
 ... **SBR**
 ... **00**
 ... **04** Utilisation Module maths MU-10

Bien sûr, ces trois méthodes ne donnent pas le même résultat en terme de nombre de pas et en temps d'exécution :

	Méthode 1		Méthode 2		Méthode 3	
mémoires	nb pas	temps	nb pas	temps	nb pas	temps
00 à 09	10	3,5 s	7	0,4 s	6	2,4 s
00 à 29	11	10,5 s	7	0,4 s	7	7 s

La 1^{ère} méthode qui paraît la plus judicieuse en terme de programmation est pourtant la plus coûteuse en terme de pas ainsi qu'en terme de temps. Elle reste pourtant la plus utilisée.

La 2^{ème} méthode est la gagnante en temps d'exécution mais n'offre pas de compatibilité entre les **TI58/58C** et la **TI59** car les définitions de groupes de mémoire ne sont pas les mêmes (Voir **OP 17**).

La 3^{ème} méthode, peu souvent employée, est un bon compromis et mériterait plus d'attention.

Séquence répétitive

Dans un programme, la présence de suite d'instructions similaires à plusieurs endroits du code est assez fréquente.

La question qui se pose alors est de savoir si il est judicieux, ou pas, de transformer cette séquence répétitive en procédure avec appel à chaque fois que nécessaire.

Bien que certaines méthodes s'appuient sur une modularité à outrance, le propos n'est pas de systématiser la démarche mais plutôt d'estimer quand elle peut être bénéfique.

Les exemples suivants reposent sur le principe de 3 instructions répétées à 3 endroits d'un même programme. (**2nd** **Int** **STO** **0** **1**)

Solution 1 :

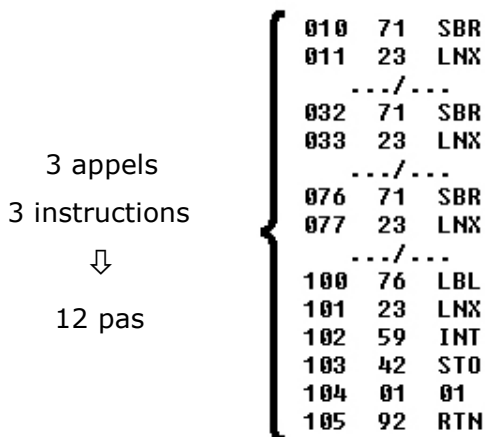
Rédaction de la séquence d'instruction autant de fois que nécessaire.

3 séquences	{	010	59	INT
		011	42	STO
		012	01	01
		.../...		
3 instructions		032	59	INT
		033	42	STO
		034	01	01
		.../...		
↓		076	59	INT
		077	42	STO
9 pas		078	01	01

nb pas = nb séquences * nb instructions

Solution 2 :

Appel d'une procédure par adressage relatif (*label*).



$$\text{nb pas} = (\text{nb appels} * 2) + (\text{nb instructions} + 3)$$

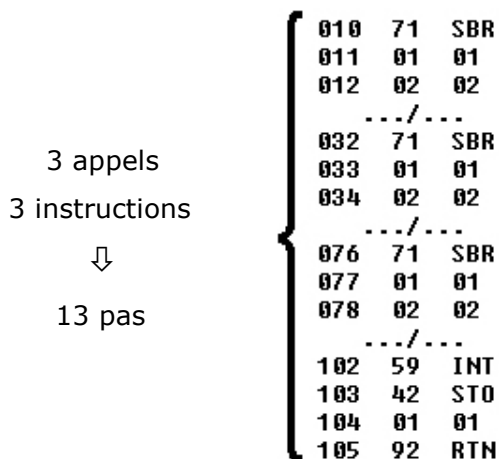
Le tableau récapitulatif suivant nous permet de déterminer à partir de combien d'instructions et de combien d'appels une économie de pas substantielle peut être faite.

		Appels		1		2		3		4		5		6		7		8	
		Solution	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	
Instructions	1	1	6	2	8	3	10	4	12	5	14	6	16	7	18	8	20		
	2	2	7	4	9	6	11	8	13	10	15	12	17	14	19	16	21		
	3	3	8	6	10	9	12	12	14	15	16	18	18	21	20	24	22		
	4	4	9	8	11	12	13	16	15	20	17	24	19	28	21	32	23		
	5	5	10	10	12	15	14	20	16	25	18	30	20	35	22	40	24		
	6	6	11	12	13	18	15	24	17	30	19	36	21	42	23	48	25		
	7	7	12	14	14	21	16	28	18	35	20	42	22	49	24	56	26		
	8	8	13	16	15	24	17	32	19	40	21	48	23	56	25	64	27		
	9	9	14	18	16	27	18	36	20	45	22	54	24	63	26	72	28		
	10	10	15	20	17	30	19	40	21	50	23	60	25	70	27	80	29		

Nous aurions aussi pu étudier une troisième solution...

Solution 3 :

Appel d'une procédure par adressage absolu (*adresse instruction*)



$$\text{nb pas} = (\text{nb appels} * 3) + (\text{nb instructions} + 1)$$

Test de boucle

RST est la commande qui ramène le pointeur d'exécution en début de mémoire programme.

En fait 3 possibilités permettent de revenir en début de partition :

- **RST**, bien sûr, mais cette instruction remet aussi les drapeaux à zéro ainsi que les adresses de retour des sous-programmes,
- **GTO** **0** **0** **0**,
- **GTO** *label*.

Trois simples petits programmes peuvent aider à comparer les performances de chaque cas.

```
000 85 +
001 01 01
002 81 RST
```

Cas N° 1

```
000 85 +
001 01 01
002 61 GTO
003 00 00
004 00 00
```

Cas N° 2

```
000 76 LBL
001 23 LNX
002 85 +
003 01 01
004 61 GTO
005 23 LNX
```

Cas N° 3

Chaque exécution est lancée par **RST R/S** puis stoppée par **R/S** après 60 secondes.

Comptage pendant 1 mn			
	Résultat (+1)	Nb pas	Ratio
Cas N°1	538	3	179.33
Cas N°2	299	5	59.80
Cas N°3	350	6	58.33

Le test semble prouver, en dehors de **RST** (Cas N° 1), que l'adressage relatif (Cas N° 3) serait sensiblement plus performant que l'adressage absolu (Cas N° 2).

Par contre, **RST** paraît intéressant, alors que peu usité, car économique en terme de pas cette instruction est des plus rapides et mériterait un peu d'attention.

Appel de procédure

Le type d'adressage, absolu (adresse) ou relatif (label), est utilisable avec toutes les instructions de branchement conditionnelles ou directes.

Le test de boucle précédemment exécuté tendrait à prouver que l'adressage relatif serait sensiblement plus performant que l'adressage absolu, mais d'autres comparaisons amènent à affiner ce jugement.

Pour chaque type d'adressage, 3 cas doivent nous permettre d'en savoir plus :

- N°1 : Appel d'une procédure en début de mémoire programme,
- N°2 : Appel d'une procédure en milieu de mémoire programme,
- N°3 : Appel d'une procédure en fin de mémoire programme.

1) adressage relatif :

```
000 71 SBR
001 45 YX
002 81 RST
003 76 LBL
004 45 YX
005 85 +
006 01 01
007 92 RTN
```

Cas N°1

```
000 71 SBR
001 45 YX
002 81 RST
.../...
235 76 LBL
236 45 YX
237 85 +
238 01 01
239 92 RTN
```

Cas N°2

```
000 71 SBR
001 45 YX
002 81 RST
.../...
475 76 LBL
476 45 YX
477 85 +
478 01 01
479 92 RTN
```

Cas N°3

2) adressage absolu :

```

000 71 SBR
001 00 00
002 04 04
003 81 RST
004 85 +
005 01 1
006 92 RTN

```

Cas N°1

```

000 71 SBR
001 02 02
002 37 37
003 81 RST
.../...
237 85 +
238 01 1
239 92 RTN

```

Cas N°2

```

000 71 SBR
001 04 04
002 77 77
003 81 RST
.../...
477 85 +
478 01 1
479 92 RTN

```

Cas N°3

Chaque programme est exécuté (**RST R/S**) puis interrompu par **R/S** après 60 secondes.

		Comptage pendant 1 mn		
		Cas N° 1	Cas N° 2	Cas N° 3
Adressage	relatif	224	66	32
	absolu	208	196	186

Ces programmes nous prouvent que les deux types d'adressage sont concurrentiels pour les adresses basses mais que l'adressage absolu est plus rapide pour les adresses hautes.

La calculatrice et ses fonctions statistiques peuvent servir à faire une analyse de tendance :

1) Pour l'adressage absolu, introduisons notre échantillon :

Adresse pas **X/T** comptage **STA**

4	Σ+	2	0	8	2nd	Σ+		
2	3	7	Σ+	1	9	6	2nd	Σ+
4	7	7	Σ+	1	8	6	2nd	Σ+

Nous pouvons calculer différentes valeurs y (*comptage*) de la droite de régression en introduisant différentes valeurs x (*adresse pas*) suivies de **OP 14**.

9 2nd **Op** 1 4 donne 207,...

5 9 2nd **Op** 1 4 donne 204,...

1 0 9 2nd **Op** 1 4 donne 202,...

et ainsi de suite jusqu'à 459.

2) Pour l'adressage relatif, introduisons notre échantillon :

Adresse pas **X/T** comptage **STA**

4	Σ+	2	2	4	2nd	Σ+	
2	3	7	Σ+	6	6	2nd	Σ+
4	7	7	Σ+	3	2	2nd	Σ+

Nous pouvons calculer différentes valeurs y (*comptage*) de la droite de régression en introduisant différentes valeurs x (*adresse pas*) suivies de **OP 14**.

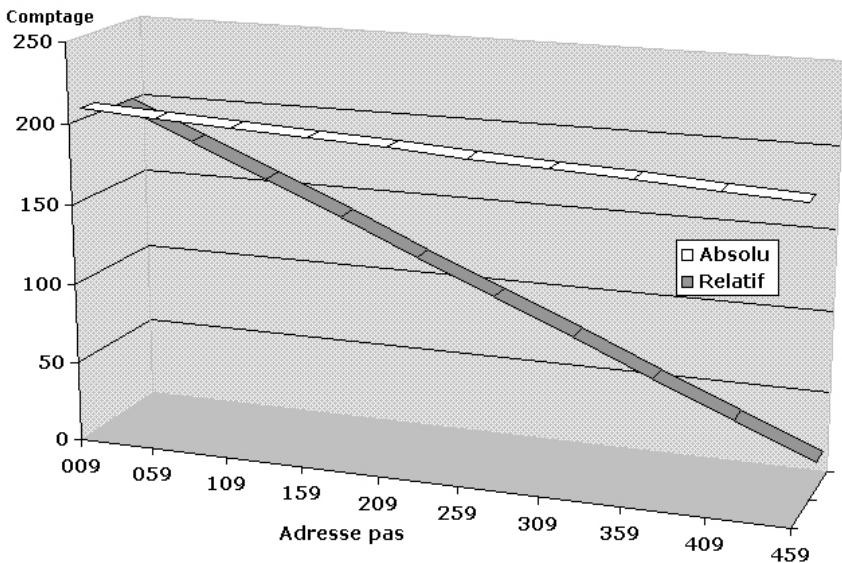
9 2nd Op 1 4 donne 200,...

5 9 2nd Op 1 4 donne 180,...

1 0 9 2nd Op 1 4 donne 160,...

et ainsi de suite jusqu'à 459.

Nous obtenons les données suivantes :



Ce graphique confirme la performance de l'adressage absolu.

Les données

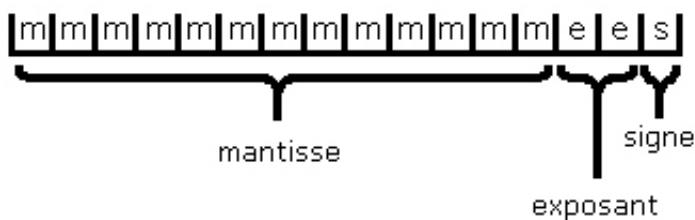
Structure des données

Les données sont affichées sur 10 chiffres avec éventuellement le signe moins.

Dans le cas d'affichage en notation scientifique (**EE**) la mantisse est affichée sur 8 chiffres et l'exposant sur 2 chiffres avec affichage éventuel des signes moins (mantisse et/ou exposant).

Dans tous les cas, la gestion interne des registres reste la même : la mantisse sur 13 caractères, l'exposant sur 2 caractères et 1 caractère pour exprimer les signes.

Soit un total de 16 caractères (ou 2 octets)



Valeur signe	Signes	
	Mantisse	Exposant
0	+	+
2	-	+
4	+	-
6	-	-

Analyse des données

La mémoire de la calculatrice est partagée entre programme et données. Ce partage est modifiable (**2nd** **Op** **1** **7**) pour répartir la mémoire entre pas de programmes et registres de stockage des données :

OP 17	TI58/TI58C
0	479-00
1	399-09
2	319-19
3	239-29
4	159-39
5	079-49
6	000-59

En prenant comme référence la **TI58**, nous constatons que 480 pas de programmes correspondent à 60 registres.

Un registre prend ainsi la place de 8 pas.

Nous avons soit 60 registres de 16 caractères, soit 480 pas de 2 caractères : la **TI58** a donc 960 caractères de mémoire utilisables. (1920 pour la **TI59**)

Dans le cas d'une partition **TI58** « **239-29** » (**3** **2nd** **Op** **1** **7**) nous avons 480 octets disponibles pour le programme (240 pas) et 480 octets disponibles pour les données (30 registres).

Cette répartition entre programme et données nous autorise à faire une équivalence entre pas et registres (pour la **TI58**) :

- au registre 00 correspondent les pas 479, 478, 477, 476, 475, 474, 473, 472.
- au registre 59 correspondent les pas 007, 006, 005, 004, 003, 002, 001, 000.
- etc ...

Pas			Pas			Pas			Pas		
Reg	de	à	Reg	de	à	Reg	de	à	Reg	de	à
00	479	472	15	359	352	30	239	232	45	119	112
01	471	464	16	351	344	31	231	224	46	111	104
02	463	456	17	343	336	32	223	216	47	103	096
03	455	448	18	335	328	33	215	208	48	095	088
04	447	440	19	327	320	34	207	200	49	087	080
05	439	432	20	319	312	35	199	192	50	079	072
06	431	424	21	311	304	36	191	184	51	071	064
07	423	416	22	303	296	37	183	176	52	063	056
08	415	408	23	295	288	38	175	168	53	055	048
09	407	400	24	287	280	39	167	160	54	047	040
10	399	392	25	279	272	40	159	152	55	039	032
11	391	384	26	271	264	41	151	144	56	031	024
12	383	376	27	263	256	42	143	136	57	023	016
13	375	368	28	255	248	43	135	128	58	015	008
14	367	360	29	247	240	44	127	120	59	007	000

Nous pouvons vérifier par la pratique cette logique de correspondance.

En mode « calculatrice », saisissons :

Touches						Affichage	
4	2nd	Op	1	7		159.39	Changement partition
2nd	II					3.14159265	PI
STO	3	0					mémorise dans registre 30
3	2nd	Op	1	7		239.29	Changement partition
GTO	2	3	9	LRN		239 31	mode programmation

Analysons les pas, à « reculons » :

L'afficheur nous donne **239 31** puis ...

- **BST** nous donne **238 41**
- **BST** nous donne **237 59**
- **BST** nous donne **236 26**
- **BST** nous donne **235 53**
- **BST** nous donne **234 59**
- **BST** nous donne **233 00**
- **BST** nous donne **232 00**

Soit :

	Mantisse										Exp.	S.				
Reg. 30	3	1	4	1	5	9	2	6	5	3	5	9	0	0	0	0
Pas	239	238	237	236	235	234	233	232								

Registres internes

Les registres internes manipulables avec l'instruction cachée **HIR** sont utilisés par la pile AOS dont il faut comprendre le fonctionnement afin d'éviter les conflits entre une utilisation personnelle de ces registres et la gestion faite par la calculatrice de ces même registres.

L'opération suivante utilise toute la pile, donc tous les registres internes :

$$2 \times (8 - (90 / (3 * (9 - (1 + (45 / (3 * 5))))))) =$$

Une analyse de ces registres à l'aide d'un programme (voir page suivante) nous donne :

2.	HIR11
8.	HIR12
90.	HIR13
3.	HIR14
9.	HIR15
1.	HIR16
45.	HIR17
3.	HIR18

Soit toutes les opérandes saisies jusqu'à la première parenthèse fermante.

000	76	LBL
001	11	A
002	02	2
003	65	*
004	53	(
005	08	8
006	75	-
007	53	(
008	09	9
009	00	0
010	55	/
011	53	(
012	03	3
013	65	*
014	53	(
015	09	9
016	75	-
017	53	(
018	01	1
019	85	+
020	53	(
021	04	4
022	05	5
023	55	/
024	53	(
025	03	3
026	65	*
027	05	5
028	54)
029	54)
030	54)
031	54)
032	54)
033	54)
034	54)
035	95	=
036	91	R/S
037	76	LBL
038	12	B
039	82	HIR
040	11	11

041	42	STO
042	01	01
043	82	HIR
044	12	12
045	42	STO
046	02	02
047	82	HIR
048	13	13
049	42	STO
050	03	03
051	82	HIR
052	14	14
053	42	STO
054	04	04
055	82	HIR
056	15	15
057	42	STO
058	05	05
059	82	HIR
060	16	16
061	42	STO
062	06	06
063	82	HIR
064	17	17
065	42	STO
066	07	07
067	82	HIR
068	18	18
069	42	STO
070	08	08
071	71	SBR
072	69	OP
073	00	0
074	02	2
075	69	OP
076	04	04
077	43	RCL
078	01	01
079	69	OP
080	06	06
081	71	SBR

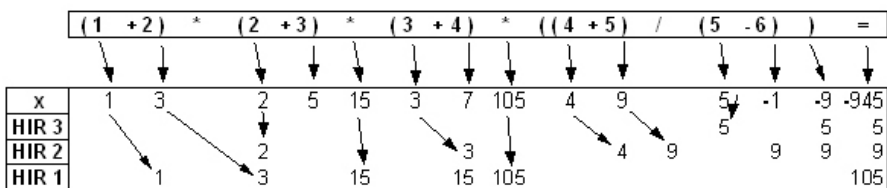
082	69	OP
083	00	0
084	03	3
085	69	OP
086	04	04
087	43	RCL
088	02	02
089	69	OP
090	06	06
091	71	SBR
092	69	OP
093	00	0
094	04	4
095	69	OP
096	04	04
097	43	RCL
098	03	03
099	69	OP
100	06	06
101	71	SBR
102	69	OP
103	00	0
104	05	5
105	69	OP
106	04	04
107	43	RCL
108	04	04
109	69	OP
110	06	06
111	71	SBR
112	69	OP
113	00	0
114	06	6
115	00	0
116	69	OP
117	04	04
118	43	RCL
119	05	05
120	69	OP
121	06	06
122	71	SBR

123	69	OP
124	00	0
125	07	7
126	69	OP
127	04	04
128	43	RCL
129	06	06
130	69	OP
131	06	06
132	71	SBR
133	69	OP
134	01	1
135	00	0
136	69	OP
137	04	04
138	43	RCL
139	07	07
140	69	OP
141	06	06
142	71	SBR
143	69	OP
144	01	1
145	01	1
146	69	OP
147	04	04
148	43	RCL
149	08	08
150	69	OP
151	06	06
152	91	R/S
153	76	LBL
154	69	OP
155	02	2
156	03	3
157	02	2
158	04	4
159	03	3
160	05	5
161	00	0
162	02	2
163	92	RTN

La pile AOS fonctionne de la façon suivante :

- Un nombre suivi d'un opérateur stocke le registre d'affichage **x** (nombre précédent ou résultat intermédiaire) dans le registre **HIR** de rang **r**,
- Un opérateur ou une parenthèse ouvrante ajoute 1 au rang **r**,
- Une parenthèse fermante exécute le dernier opérateur entre le registre d'affichage **x** et le registre **HIR** de rang **r**, met le résultat dans le registre d'affichage **x** puis soustrait 1 du rang **r**.

Exemple :



Tout résultat intermédiaire intervient à l'affichage avant d'être mis en réserve dans la pile AOS.

Les registres internes **HIR** sont aussi utilisés par les fonctions d'impression alphanumérique.

Si, en mode « calculatrice », nous saisissons :

6 4 6 4 6 4 6 4 6 4 **OP 01**

3 6 3 6 3 6 3 6 3 6 **OP 02**

5 2 5 2 5 2 5 2 5 2 **OP 03**

7 7 7 7 7 7 7 7 7 7 **OP 04**

OP 05 nous donne :

=====SSSSSrrrrrrzzzzz

Nous constatons le contenu des registres internes 5 à 8 :

- **HIR 15** donne .0064646465 (6464646464000034 en interne)
- **HIR 16** donne .0036363636 (3636363636000034 en interne)
- **HIR 17** donne .0052525253 (5252525252000034 en interne)
- **HIR 18** donne .0077777778 (7777777777000034 en interne)

Voilà pourquoi le programme de la page 106 récupère les registres **HIR** pour les stocker dans les registres de données 00 à 08 avant d'utiliser les fonctions d'impression OP : afin d'éviter les conflits entre une utilisation personnelle de ces registres et la gestion faite par la calculatrice de ces même registres.

Comment pratiquer ?

Les calculatrices **TI59/58/58C** nécessaires à la pratique du langage LMS ne sont plus commercialisées depuis de nombreuses années.

Bien qu'il soit parfois possible d'en trouver d'occasion lors de brocantes ou sites internet d'enchères, ces opportunités sont plutôt rares et l'état des machines ainsi dénichées n'est pas vraiment garanti, le clavier ayant des tendances aux « rebonds » et les accus étant souvent défectueux.

Heureusement la passion des « aficionados » a perduré au fil des ans et internet offre divers sites proposant des informations intéressantes, les manuels et autres documentations mais surtout des solutions de substitution : des émulateurs fonctionnant sur des PC ou tablettes (MS Dos, Windows, Android, Pocket PC).

Un émulateur de **TI59/58/58C** sur plateforme Windows est proposé sur un site web entièrement consacré à ces calculatrices, et il est désormais possible de s'adonner aux plaisirs de ce langage en téléchargeant ce logiciel gratuit sur

<http://ti58c.ift.cx>

Ce site référence la plupart des logiciels d'émulation disponibles et donne aussi les principaux liens vers les autres sites dédiés à ces calculatrices.

Sommaire

Introduction	3
Premier programme	5
Premiers pas	7
Introduction du programme.....	9
Promenons nous.....	11
Premier test	13
Stockage en mémoire.....	17
Impression.....	20
Programme complet	23
Le langage.....	31
Programmation.....	34
Touches complémentaires	35
Entrée des données.....	38
Les opérations arithmétiques.....	39
Effacement.....	40
Racines et puissances.....	41
Fonctions mathématiques	42
Trigonométrie.....	44
Impression.....	46
Options d’affichage.....	50
Gestion des données	54
Branchements	57
Statistiques.....	66

Touches de fonctions	69
Lecture / écriture	70
Modules de librairie	72
Opérations spéciales.....	74
Autres fonctions	76
L'instruction cachée.....	77
Tableau récapitulatif des instructions.....	81
Tests Comparatifs.....	83
Remise à zéro des registres.....	86
Séquence répétitive.....	89
Test de boucle	92
Appel de procédure	94
Les données	99
Structure des données.....	101
Analyse des données	102
Registres internes	105
Comment pratiquer ?.....	109

Index

I**|X|** 38

1**1/X** 24, 27, 29, 42, 44

2**2nd** 35

A**ADV** 20, 46**AVR** 35, 67, 77

B**BST** 34

C**CE** 40**CLR** 23, 24, 27, 29, 30, 40**CMS** 40**COS** 23, 25, 35, 44**CP** 10, 34, 40

D**DEG** 43, 44**DEL** 34, 76**DMS** 35, 45, 77**DSZ** 35, 36, 37, 64, 65

E**EE** 35, 51**ENG** 35, 51**EQ** 35, 36, 37, 60, 61, 62, 69**EXC** 36, 55

F**FIX**... 24, 27, 30, 35, 36, 37, 53

G**GE** ... 23, 26, 29, 35, 36, 37, 61,
62, 69**GRD** 43, 44**GTO** . 36, 57, 58, 60, 61, 63, 64,
69

H**HIR** .. 77, 79, 80, 105, 107, 108

I

IFF 35, 36, 37, 63, 64
IND . 35, 36, 37, 53, 59, 61, 62,
 63, 64, 65, 75
INS.....13, 34
INT.....35, 38
INV . 20, 23, 24, 26, 27, 29, 30,
 35, 36, 37, 38, 41, 42, 43,
 44, 45, 46, 51, 53, 55, 56,
 61, 62, 63, 64, 65, 67, 71, 80

L

LBL.. 13, 14, 18, 23, 24, 25, 26,
 27, 29, 30, 35, 57
LNx . 23, 24, 25, 29, 30, 35, 42,
 60
LOG . 23, 24, 27, 29, 30, 35, 42,
 59
LRN 1, 34
LST..... 12, 20, 35, 46

N

NOP..... 76

O

OP 00 21, 22, 46, 75, 77
OP 01 ... 20, 21, 22, 46, 74, 77,
 108
OP 02 ... 20, 21, 22, 46, 77, 108
OP 03 20, 21, 46, 77, 108
OP 04 ... 20, 21, 22, 47, 77, 108
OP 05 20, 21, 22, 47, 108
OP 06 20, 22, 47
OP 07 20, 47
OP 08 20, 49, 74

OP 09 73, 74
OP 10 38, 74
OP 11 67, 74, 77
OP 12 68, 77
OP 13 68, 77
OP 14 68, 77, 96, 97
OP 15 68, 74, 77
OP 16 17, 40, 74, 75
OP 17 17, 40, 74, 88, 102
OP 18 75
OP 19 75
OP 2n 56
OP 3n 56
OP 40 75

P

P/R 35, 42, 43, 77
PAU 76
PGM 36, 73
PRD 35, 36, 55, 80
PRT 20, 24, 27, 30, 46

R

R/S . 14, 18, 23, 24, 27, 28, 29,
 30, 59, 60
RAD 43, 44
RCL . 18, 23, 27, 29, 30, 36, 42,
 54, 79
RST 60
RTN.. 23, 24, 25, 27, 30, 58, 59

S

SBR . 23, 25, 29, 30, 35, 36, 37,
 58, 59, 69
SIN 23, 25, 35, 44
SQR 41
SST 34

STA 35, 67, 77, 96
STF..... 35, 36, 37, 63
STO 18, 23, 29, 36, 54, 79
SUM.. 35, 36, 54, 55, 56, 79, 80

T

TAN 23, 25, 35, 44

W

WRI. 23, 24, 25, 26, 27, 35, 70,
71

X

X/T 23, 26, 29, 54, 96
X2 14, 23, 29, 41

Y

Yx 35, 41