Microprogramme Mic 1

	Main1	PC = PC + 1; fetch; goto (MBR)	(MBR contient code d'opération)extrait octet suivant; branchement
1	nop1	goto Main1	Ne fait rien
2	iadd1	MAR = SP = SP – 1; rd	Lit le 2ème mot de la pile
3	iadd2	H = TOS	H = sommet de pile
4	iadd3	MDR = TOS = MDR + H; wr; goto Main1	Additionne les 2 mots,actualise TOS ; écrit nouveau sommet
5	isub1	MAR = SP = SP – 1; rd	Lit 2ème mot de la pile
6	isub2	H = TOS	H =sommet de pile
7	isub3	MDR = TOS = MDR – H; wr; goto Main1	Soustrait, actualise TOS ;écrit nouveau sommet
8	iand1	MAR = SP = SP – 1; rd	Lit 2ème mot de la pile
9	iand2	H = TOS	H =sommet de pile
10	iand3	MDR = TOS = MDR AND H; wr; goto Main1	Effectue AND, actualise TOS ;écrit nouveau sommet
11	ior1	MAR = SP = SP – 1; rd	Lit 2ème mot de la pile
12	ior2	H = TOS	H =sommet de pile
13	ior3	MDR = TOS = MDR OR H; wr; goto Main1	Effectue OR, actualise TOS ;écrit nouveau sommet
14	dup1	MAR = SP = SP + 1	MAR et SP =futur sommet de pile
15	dup2	MDR = TOS; wr; goto Main1	Ecrit nouveau sommet
16	pop1	MAR = SP = SP - 1; rd	Lit 2ème mot de pile
17	pop2		Attend disponibilité du 2ème mot
18	pop3	TOS = MDR; goto Main1	Actualise TOS avec nouveau sommet de pile
19	swap1	MAR = SP - 1; rd	MAR =position 2ème mot de pile;lit le 2ème mot
20	swap2	MAR = SP	MAR =sommet de pile
21	swap3	H = MDR; wr	H =2ème mot; écrit nouveau sommet
22	swap4	MDR = TOS	MDR =ancien sommet
23	swap5	MAR = SP - 1; wr	Ecrit ancien sommet en 2ème position de pile
24	swap6	TOS = H; goto Main1	Actualise TOS avec nouveau sommet
25	bipush1	SP = MAR = SP + 1	(MBR =octet à empiler) MAR et SP =nouveau sommet
26	bipush2	PC = PC + 1; fetch	Prépare PC ; extrait prochain code d'opération
27	bipush3	MDR = TOS = MBR; wr; goto Main1	TOS et MDR =octet signé étendu; écrit nouveau sommet
28	iload1	H = LV	(MBR =index)H =base des variables locales
29	iload2	MAR = MBRU + H; rd	MAR =adresse variable locale;lit variable
30	iload3	MAR=SP=SP+1	MAR et SP=nouveau sommet
31	iload4	PC = PC + 1; fetch; wr	Extrait prochain code d'opération; écrit nouveau sommet
32	iload5	TOS = MDR; goto Main1	Actualise TOS
33	istore1	H= LV	(MBR = index) H = base des variables locales
34	istore2	MAR = MBRU + H	MAR = adresse variable locale
35	istore3	MDR = TOS; wr	MDR = sommet de pile ; écrit dans variable
36	istore4	SP = MAR = SP – 1; rd	Lit 2e mot de pile
37	istore5	PC = PC + 1; fetch	Extrait code d'opération suivant
38	istore6	TOS = MDR; goto Main1	Actualise TOS avec nouveau sommet
39	wide1	PC = PC + 1 ; fetch ;goto (MBR OR 0x100)	Extrait octet suivant (opérande ou code d'o p.) Branchement avec QQ bit de MPC active
40	wide_iload1	PC = PC + 1 ; fetch	(MBR contient 1er octet index) Extrait 2e octet index
41	wide_iload2	H = MBRU <<8	H = 1er octet étendu non signé décalé de 8 bits
42	wide_iload3	H = MBRU OR H	H = index de 16 bits de la variable locale
43	wide_iload4	MAR = LV + H; rd; goto iload3	MAR = adresse var. loc.a empiler ; lit variable
44	wide_istore1	PC = PC + 1; fetch	(MBR :1er octet index)Extrait 2e octet index
45	wide_istore2	H = MBRU <<8	H = 1er octet étendu non signé décalé de 8 bits
46	wide_istore3	H = MBRU OR H	H = index de 16 bits de la variable locale
47	wide_istore4	MAR = LV + H; goto istore3	MAR = adresse variable locale cible
48	ldc_w1	PC = PC + 1; fetch	(MBR :1er octet index) Extrait 2e octet index
49	ldc_w2	H = MBRU <<8	H = 1er octet étendu non signé décalé de 8 bits
50	ldc_w3	H = MBRU OR H	H = index de 16 bits du pool de constantes
51	ldc_w4	MAR = H + CPP; rd; goto iload3	MAR = adresse constante dans le pool;lit constante
52	iinc1	H = LV	(MBR =index);H =base variables locales
53	iihc2	MAR = MBRU + H; rd	MAR =base+index; lit variable
54	iinc3	PC = PC + 1; fetch	Extrait constante à additionner
55	iinc4	H = MDR	H =variable lue
56	iinc5	PC = PC + 1; fetch	Extrait nouveau code d'opération
57	iinc6	MDR = MBR + H; wr; goto Main1	MDR = constante+variable;actualise variable
58	goto1	OPC = PC - 1	Préserve l'adresse du code d'opération actuel
59	goto2	PC = PC + 1; fetch	(MBR = 1er octet index) Extrait 2ème octet index
60	goto3	H= MBR <<8	H = 1er octet signé décalé de 8 bits
61	goto4	H = MBRU OR H	H =offset 16 bits de branchement
62	goto5	PC = OPC + H; fetch	PC =base constantes+index;extrait code d'opération
63	goto6	goto Mainl	Attente disponibilité du code d'opération dans MBR
64	iflt1	MAR = SP = SP – 1; rd	MAR et SP =futur sommet;lit 2ème mot de pile
65	iflt2	OPC = TOS	Préserve temporairement sommet de pile dans OPC
66	iflt3	TOS = MDR	Actualise TOS avec nouveau sommet
67	iflt4	N = OPC; if (N) goto T; else goto F	Teste;branchement vers T si N activé, sinon vers F
68	ifeq1	MAR = SP = SP – 1; rd	Lit 2ème mot de la pile
69	ifeq2	OPC = TOS	Préserve temporairement sommet de pile dans OPC
70	ifeq3	TOS = MDR	Actualise TOS avec nouveau sommet
71	ifeq4	Z = OPC; if (Z) goto T; else goto F	Teste; branchement vers T si Z activé,sinon vers F
72	if_icmpeq1	MAR = SP = SP – 1 ; rd	Lit 2e mot de pile

```
73  if_icmpeq2        MAR = SP = SP — 1                    MAR et SP = futur sommet de pile
74  if_icmpeq3        H = MDR; rd                          H = mot lu ; lit nouveau sommet de pile
75  if_icmpeq4        OPC = TOS                            Préserve temporairement ancien sommet dans OPC
76  if_icmpeq5        TOS = MDR                            Actualise TOS avec nouveau sommet
77  if_icmpeq6        Z = OPC — H; if (Z) goto T; else goto F   Teste;branchement vers T si Z activé,sinon vers F

78  T   OPC = PC — 1; goto goto2                           Comme pour goto1,utile pour adresse cible

79  F   PC = PC + 1                                        Saut 1er octet offset, PC pointe sur 2ème octet
80  F2  PC = PC + 1; fetch                                 PC pointe sur prochain code d'opération ;extrait code
81  F3  goto Main1                                         Attente de disponibilité du code dans MBR

82  invokevirtual1    PC = PC + 1 ; fetch                  (MBR = le 1er 0ctet index) Extrait 2ème octet index
83  invokevirtual2    H = MBRU <<8                         H = 1er octet non signé décalé de 8 bits
84  invokevirtual3    H = MBRU OR H                        H = offset du pointeur de la méthode appelée
85  invokevirtual4    MAR = CPP + H; rd                    Récupère pointeur vers méthode dans zone CPP
86  invokevirtual5    OPC = PC + 1                         Préserve temporairement PC de retour dans OPC
87  invokevirtual6    PC = MDR; fetch                      PC =adresse méthode; extrait 1er octet du nombre param;
88  invokevirtual7    PC = PC + 1; fetch                   Extrait 2ème octet du nombre de parametres
89  invokevirtual8    H = MBRU <<8                         H =1er octet non signe decale de 8bits
90  invokevirtual9    H = MBRU OR H                        H =nombre de parametres
91  invokevirtual10   PC = PC + 1; fetch                   Extrait ler octet du nombre de variables locales
92  invokevirtual11   TOS = SP — H                         TOS = adresse de OBJREF — 1
93  invokevirtual12   TOS = MAR = TOS + 1                  TOS = adresse de OBJREF (nouveau LV)
94  invokevirtual13   PC = PC + 1; fetch                   Extrait 2ème octet du nombre de variables locales
95  invokevirtual14   H = MBRU <<8                         H =1er octet non signe décalé de 8 bits
96  invokevirtual15   H = MBRU OR H                        H =nombre de variables locales
97  invokevirtual16   MDR = SP + H + 1; wr                 Ecrase OBJREF avec pointeur de liaison
98  invokevirtual17   MAR = SP = MDR                       MAR et SP =adresse contenant ancien PC
99  invokevirtual18   MDR = OPC; wr                        MDR =ancien PC ;ecrit au dessus variables locales
100 invokevirtual19   MAR = SP = SP + 1                    Prépare MAR et SP pour stocker ancien LV
101 invokevirtual2O   MDR = LV; wr                         Ecrit ancien LV au—dessus ancien PC préservé
102 invokevirtual21   PC = PC + 1; fetch                   Extrait premier code d'opération de méthode invoquée
103 invokevirtual22   LV = TOS; goto Main1                 Actualise LV avec nouvelle base variables locales

104 ireturn1          MAR = SP = LV; rd                    Prépare SP et MAR pour récupération pointeur de liaison
105 ireturn2                                               Attend la disponibilité du pointeur
106 ireturn3          LV = MAR = MDR; rd                   LV et MAR =pointeur de liaison; lit ancien PC
107 ireturn4          MAR = LV + 1                         Prépare MAR pour lire ancien LV
108 ireturn5          PC = MDR; rd; fetch                  Restauration ancien PC ;lit ancien LV;  extrait prochain code d'opération
109 ireturn6          MAR = SP                             Prépare MAR pour écrire sommet de pile
110 ireturn7          LV = MDR                             Restaure ancien LV
111 ireturn8          MDR = TOS; wr; goto Main1            Ecrit valeur retournée sursommet de pile original
```

Instructions possibles dans l'ALU

DEST = H	DEST = $\overline{\text{SOURCE}}$	DEST = $H+1$	DEST = SOURCE − 1	DEST = H **OR** SOURCE
DEST = SOURCE	DEST = H + SOURCE	DEST = SOURCE + 1	DEST = $-H$	DEST = 0
DEST = $\overline{H}$	DEST = H + SOURCE + 1	DEST = SOURCE − H	DEST = H **AND** SOURCE	DEST = 1
				DEST = −1