```
! Variante avec prise en compte des interruptions

!*********************************************************************!
!  init                                                               !
!  PC <- 0  // I <- 1                                                 !
!*********************************************************************!

init:       pc = 0  seti

!*********************************************************************!
!  fetch                                                              !
!  Rinstr_fort (ir)            <- Mem[pc] // pc ++                    !
!  Rinstr_faible (mb // mlk2)  <- Mem[pc] // pc ++                    !
!                                                                     !
! 1er  octet (poids fort : désigne opération) dans ir                 !
! 2ème octet oooo dddd dans mb et mk2                                 !
!                                                                     !
!*********************************************************************!

fetch:      pc = 0 +pc ema
            read
            read  pc = pc +1
            ir = mb

            pc = 0 +pc ema
            read
            read  pc = pc +1
            mk2 = mb

            j7   branch_autres      ! 1x : branchement ou autres instr

!*********************************************************************!
! Section commune à mémoire et calcul                                 !
!                                                                     !
! Mettre l'opérande reg_oooo ou #oooo dans mb                         !
!*********************************************************************!
            j4 oooo_imm

            ! reg_oooo dans mk2 poids forts : lire reg[MK2]
oooo_reg:   mb = labus
            jp  oooo_lu

            ! #oooo dans mk2 poids forts    : prendre mb = mk2>> 4
oooo_imm:   mk1 = shr(mk2)
            mk1 = shr(mk1)
            mk1 = shr(mk1)
            mb  = shr(mk1)

oooo_lu:   j6   memoire             ! 01 : load ou store

!*********************************************************************!
! Gestion des instructions de calcul                                  !
!                                                                     !
! Format : code_op8  oooo rrrr                                        !
!                    o : opérande droit = ual_gauche (reg ou #4b)     !
!                    r : opérande gauche / résultat  ual droit        !
!                                                                     !
!           76 5 4       3  2  10                                     !
! Code_op8: 00 S # sansbar /+1 ual                                    !
!                        0  0  00  (+)     /o + d + 1  sub            !
!                        0  0  01  (g)     /o + 1      ---            !
!                        0  0  10  (&)     /o & d + 1  ---            !
!                        0  0  11  (^)     /o ^ d + 1  ---            !
!                        0  1  00  (+)     /o + d      ---            !
!                        0  1  01  (g)     /o          mvn            !
!                        0  1  10  (&)     /o & d      bic            !
!                        0  1  11  (^)     /o ^ d      xnor           !
!                        1  0  00  (+)      o + d + 1  ---            !
!                        1  0  01  (g)      o + 1      ---            !
```



```
!                        1  0  10  (&)      o & d + 1  ---            !
!                        1  0  11  (^)      o ^ d + 1  ---            !
!                        1  1  00  (+)      o + d      add            !
!                        1  1  01  (g)      o          mov            !
!                        1  1  10  (&)      o & d      and            !
!                        1  1  11  (^)      o ^ d      xor            !
!                                                                     !
!  Sélection opération :   2 bits de poids faible de mk2 (lual)      !

!  Lecture   reg_oooo  :   4 bits de poids fort de mk2 (labus)       !

!  Lecture   reg_dddd  :   4 bits de poids faible de mk1 (lbbus)     !
!  Ecriture  reg_dddd  :   4 bits de poids fort de mk1 (lcbus)       !
!*********************************************************************!


!1) Mettre xxxx dddd dans mk1  (pour mb = mb oper lbbus)
calcul:     mk1 = mk2

!2)  ne pas inverser si ir3 == 1
testbar:    j3  testplusun
            mb = mb xor ff

!3)  ne pas ajouter 1 si ir2 == 1
testplusun: j2 calculer
            mb = mb + 1

!4)  calculer mb = mb lual lbbus avec ou sans les flags
calculer:   mk2 = ir
            j5  avecs

sanss:      mb = mb lual lbbus
            jp ranger

avecs:      mb = mb lual lbbus majflag

!5)  lcbus = mb   : mettre dddd dans mk1 poids forts (<<4)

ranger:     mk1 = shl (mk1 + mk1)
            mk1 = shl (mk1 + mk1)
            lcbus = mb
            jp fetch

!*********************************************************************!
! Gestion des instructions d'accès à la mémoire : load et store       !
!                                                                     !
! load/store   [reg, reg_ou_#4bits]                                   !
!                                                                     !
!             ir        mk2                                           !
! Format : code_op8  oooo gggg                                        !
!          adresse = reg_gggg + oooo_reg ou #oooo                     !
!                                                                     !
!           76 5 4  3210                                              !
! Code_op8: 01 E #  rval                                              !
!                                                                     !
!  Lecture   oooo  :   déjà fait, dans mb                             !
!  Lecture   reg_gggg  :   mk1  (lbbus)                               !
!  Ecriture  reg_dest  :   MK1  (lcbus)                               !
!*********************************************************************!
memoire:

            !  a completer ....
            !
            jp fetch



branch_autres: j6  autres
```

```
!****************************************************************!
! Gestion de b/bl cond                                           !
!                        cond parmi eq (Z==1)                    !
!                                   hs (C==1)                    !
!                                   __ (toujours)                !
!                                                                !
! Format : code_op8  deplacement_8_en_octets                     !
!                                                                !
!           76 5 4  3210                                         !
! Code_op8: 10 /l z C xxx                                        !
!              0 0 0 xxx  bl                                     !
!              1 0 0 xxx  b                                      !
!              0 1 x xxx  bleq                                   !
!              1 1 x xxx  beq                                    !
!              0 0 1 xxx  blcs                                   !
!              1 0 1 xxx  bcs                                    !
!****************************************************************!

!1) Calculer adresse de destination dans mb (deplacement dans mk2)
              mb = pc + mk2

!2) Copier adresse de retour dans mk1
              mk1 = pc

!3) tester condition de branchement

            j3   beq
            j2   bcs
            jp   brancher        ! bal : inconditionnel !

            ! amener flag à tester en bit 0
            ! C : bit 2
bcs:        mk2 = shr (f)
            mk2 = shr (mk2)
            jp test_flag

            ! Z : bit 0
beq:        mk2 = f

test_flag:  mk2 = mk2 and 1
            jz   fetch

brancher:   pc = mb
            j5   fetch           ! pas bl : terminé     !
            tir = mk1            ! bl : sauver --> lr/tir !
            jp   fetch

!****************************************************************!
! Gestion des autres instructions                                !
!                                                                !
! 7654 3210                                                      !
! 1100          mov  cpsr/spsr, reg                              !
! 1101          mov  reg, cpsr/spsr                              !
! 1111          rti                                              !
!                                                                !
! 1110 0        seti                                             !
! 1110 1        clri                                             !
!****************************************************************!

autres:      j5  instr_it
             !  ce sont les mov sur les psr
             ! on se ramène à un traitement de move
             mb = labus
             jp calcul

instr_it:    j3  clear

set:         seti
             jp fetch
```



```
clear:       clri
             jp fetch
```