SOLUTION EXERCICE 4_1 PIC32MX

PARTIE A)

Voici tout d'abord le programme en C :

```
#include <stddef.h>                      // Defines NULL
#include <stdbool.h>                     // Defines true
#include <stdlib.h>                     // Defines EXIT_FAILURE
#include <stdint.h>                     // Defines EXIT_FAILURE

// Fonction SimuPrintSymb
void SimuPrintSymb (const char *format, char val)
{
    char tmp = val;
}

// Fonction SerieSymboles
//  Affiche une série de symboles avec possibilité de
changement de ligne
void SerieSymboles (uint8_t n, char symb, bool IsEndl)
{
    uint8_t i;
    for (i=0; i < n; i++) {
        SimuPrintSymb("%c", symb);
    }
    if (IsEndl == true) {
        symb = '\n';
        SimuPrintSymb("%c", symb);
    }
}

// Fonction ComposeLigne
// Compose une ligne avec 2 séries
void ComposeLigne (uint8_t n1, char symb1,uint8_t n2, char
symb2)
{
    SerieSymboles(n1, symb1, false);
    SerieSymboles(n2, symb2, true);
}

// Fonction DispMotif
// Compose un motif de n lignes
void DispMotif (uint8_t nbli, char symb1, char symb2)
{
   uint8_t i;
   for (i=1; i <= nbli; i++) {
        ComposeLigne(i, symb1, nbli-i, symb2);
    }
}
```

```
uint16_t ExtractHeures(uint32_t nbSecTot)
{
    return (nbSecTot / 3600);
}

uint16_t ExtractMinutes(uint32_t nbSecTot, uint16_t NbH)
{
    return ( (nbSecTot - ( NbH * 3600)) / 60);
}

uint16_t ExtractSecondes(uint32_t nbSecTot, uint16_t NbH,
                                            uint16_t NbM)
{
    return ( nbSecTot - ( NbH * 3600) - (NbM *60) );
}

typedef struct {
    uint16_t NbH;
    uint16_t NbM;
    uint16_t NbS;
} S_HMS;

S_HMS ExtractHMS(uint32_t nbSecTot)
{
    S_HMS tmp;
    tmp.NbH = nbSecTot / 3600;
    tmp.NbM = (nbSecTot - ( tmp.NbH * 3600)) / 60;
    tmp.NbS = nbSecTot - ( tmp.NbH * 3600) - (tmp.NbM *60);
    return ( tmp );
}

void main() {
   uint32_t NbSecT = 3675;
   uint16_t H, M, S;
   S_HMS ResHMS;
   bool match;

   // Appel des fonctions d'extractions
   H = ExtractHeures(NbSecT);
   M = ExtractMinutes(NbSecT, H);
   S = ExtractSecondes(NbSecT, H, M);

   ResHMS = ExtractHMS(NbSecT);

   if (H = ResHMS.NbH) match = true;
   if (M = ResHMS.NbM) match = true;
   if (S = ResHMS.NbS) match = true;

   // Appel de la fonction  DispMotif
   DispMotif(5, '*', '+');
   DispMotif(7, '!', '=');
}
```

Voici le listing assembleur obtenu avec des commentaires permettant de comprendre les actions réalisées. Seul le mécanisme de passage de paramètre et d'appel des fonctions est commenté.

```
Disassembly Listing for EX4_1
Generated From:
D:/EtCoursHW/SL229_MINF/CoursPIC32/Exercices/ProjExer/EX4_1.X/
dist/default/production/EX4_1.X.production.elf
24 nov. 2014 10:00:24

---
d:/etcourshw/sl229_minf/courspic32/exercices/projexer/ex4_1.x/
main_ex4_1.c  ------------------------
1:      #include <stddef.h>              // Defines NULL
2:      #include <stdbool.h>             // Defines true
3:      #include <stdlib.h>              // Defines EXIT_FAILURE
4:     #include <stdint.h>               // Defines EXIT_FAILURE
5:
6:                     // Fonction SimuPrintSymb
7:                       void SimuPrintSymb (const char *format,
char val)
8:                     {
// ajout 4 mots de 32 bits dans stack
9D000000  27BDFFF0    ADDIU SP, SP, -16
9D000004  AFBE000C    SW S8, 12(SP)       // sauve S8 (FP)
9D000008  03A0F021    ADDU S8, SP, ZERO   // S8 = SP
9D00000C  AFC40010    SW A0, 16(S8)       // sauve A0
9D000010  00A01021    ADDU V0, A1, ZERO
9D000014  A3C20014    SB V0, 20(S8)       // A1 => val
9:                        char tmp = val;
9D000018  93C20014    LBU V0, 20(S8)
9D00001C  A3C20000    SB V0, 0(S8)
10:                    }
9D000020  03C0E821    ADDU SP, S8, ZERO   // SP = S8
9D000024  8FBE000C    LW S8, 12(SP)       // restaure S8 (FP)
9D000028  27BD0010    ADDIU SP, SP, 16    // maj SP
9D00002C  03E00008    JR RA               // retour par RA
9D000030  00000000    NOP
```

Remarque la fonction SimuPrintSymb est une fonction "leaf", elle n'appelle pas d'autre fonction.

```
11:
12:                    // Fonction SerieSymboles
13:                       // Affiche une série de symboles avec
possibilité de changement de ligne
```

```
14:                  void SerieSymboles (uint8_t n, char symb,
bool IsEndl)
15:                  {
// ajout 8 mots de 32 bits dans stack
9D000034  27BDFFE0   ADDIU SP, SP, -32
9D000038  AFBF001C   SW RA, 28(SP)      // sauve RA
9D00003C  AFBE0018   SW S8, 24(SP)      // sauve S8 (FP)
9D000040  03A0F021   ADDU S8, SP, ZERO  // S8 = SP
9D000044  00A01821   ADDU V1, A1, ZERO  // A1 2ème par => V1
9D000048  00C01021   ADDU V0, A2, ZERO  // A2 3ème par => V0
9D00004C  A3C40020   SB A0, 32(S8)      // A0 1er par => n
9D000050  A3C30024   SB V1, 36(S8)      // V1 2ème par => symb
9D000054  A3C20028   SB V0, 40(S8)      // V0 3ème P. => IsEndl
16:                      uint8_t i;
17:                      for (i=0; i < n; i++) {
9D000058  A3C00010   SB ZERO, 16(S8)
9D00005C  0B400022   J 0x9D000088
9D000060  00000000   NOP
9D00007C  93C20010   LBU V0, 16(S8)
9D000080  24420001   ADDIU V0, V0, 1
9D000084  A3C20010   SB V0, 16(S8)
9D000088  93C30010   LBU V1, 16(S8)
9D00008C  93C20020   LBU V0, 32(S8)
9D000090  0062102B   SLTU V0, V1, V0
9D000094  1440FFF3   BNE V0, ZERO, 0x9D000064
9D000098  00000000   NOP
18:                          SimuPrintSymb("%c", symb);
9D000064  83C20024   LB V0, 36(S8)      // symb => V0
9D000068  3C039D00   LUI V1, -25344
9D00006C  246407AC   ADDIU A0, V1, 1964     // A0 1er param
9D000070  00402821   ADDU A1, V0, ZERO      // A1 2ème param
9D000074  0F400000   JAL SimuPrintSymb
9D000078  00000000   NOP
19:                      }
20:                      if (IsEndl == true) {
9D00009C  93C20028   LBU V0, 40(S8)
9D0000A0  10400009   BEQ V0, ZERO, 0x9D0000C8
9D0000A4  00000000   NOP
21:                          symb = '\n';
9D0000A8  2402000A   ADDIU V0, ZERO, 10
9D0000AC  A3C20024   SB V0, 36(S8)
22:                          SimuPrintSymb("%c", symb);
9D0000B0  83C20024   LB V0, 36(S8)
9D0000B4  3C039D00   LUI V1, -25344
9D0000B8  246407AC   ADDIU A0, V1, 1964     // A0 1er param
9D0000BC  00402821   ADDU A1, V0, ZERO      // A1 2ème param
9D0000C0  0F400000   JAL SimuPrintSymb
9D0000C4  00000000   NOP
23:                      }
24:                  }
```

```
// préparation du retour de SerieSymboles
9D0000C8  03C0E821   ADDU SP, S8, ZERO       // SP = S8
9D0000CC  8FBF001C   LW RA, 28(SP)           // pop RA
9D0000D0  8FBE0018   LW S8, 24(SP)           // pop S8
9D0000D4  27BD0020   ADDIU SP, SP, 32        // maj SP
9D0000D8  03E00008   JR RA                   // retour par RA
9D0000DC  00000000   NOP
25:
26:                  // Fonction ComposeLigne
27:                  // Compose une ligne avec 2 séries
28:                       void ComposeLigne (uint8_t n1, char
symb1,uint8_t n2, char symb2)
29:                  {
// ajout 6 mots de 32 bits dans stack
9D0000E0  27BDFFE8   ADDIU SP, SP, -24
9D0000E4  AFBF0014   SW RA, 20(SP)           // push RA
9D0000E8  AFBE0010   SW S8, 16(SP)           // push S8
9D0000EC  03A0F021   ADDU S8, SP, ZERO       // S8 = SP
//  Mécanisme  de  transfert  des  valeurs  des  paramètres,  des
registres aux arguments dans le stack
9D0000F0  00804021   ADDU T0, A0, ZERO
9D0000F4  00A02021   ADDU A0, A1, ZERO
9D0000F8  00C01821   ADDU V1, A2, ZERO
9D0000FC  00E01021   ADDU V0, A3, ZERO
9D000100  A3C80018   SB T0, 24(S8)       // val P1 => n1
9D000104  A3C4001C   SB A0, 28(S8)       // val P2 => symb1
9D000108  A3C30020   SB V1, 32(S8)       // val P3 => n2
9D00010C  A3C20024   SB V0, 36(S8)       // val P4 => symb2
30:                      SerieSymboles(n1, symb1, false);
9D000110  93C30018   LBU V1, 24(S8)
9D000114  83C2001C   LB V0, 28(S8)
9D000118  00602021   ADDU A0, V1, ZERO       // n1 => A0
9D00011C  00402821   ADDU A1, V0, ZERO       // symb1 => A1
9D000120  00003021   ADDU A2, ZERO, ZERO     // A2 = 0
9D000124  0F40000D   JAL SerieSymboles
9D000128  00000000   NOP
31:                      SerieSymboles(n2, symb2, true);
9D00012C  93C30020   LBU V1, 32(S8)
9D000130  83C20024   LB V0, 36(S8)
9D000134  00602021   ADDU A0, V1, ZERO       // n2 => A0
9D000138  00402821   ADDU A1, V0, ZERO       // symb2 => A1
9D00013C  24060001   ADDIU A2, ZERO, 1       // A2 = 1
9D000140  0F40000D   JAL SerieSymboles
9D000144  00000000   NOP
32:                  }
// préparation du retour de ComposeLigne
9D000148  03C0E821   ADDU SP, S8, ZERO       // SP = S8
9D00014C  8FBF0014   LW RA, 20(SP)           // pop RA
9D000150  8FBE0010   LW S8, 16(SP)           // pop S8
9D000154  27BD0018   ADDIU SP, SP, 24        // maj SP
9D000158  03E00008   JR RA                   // retour par RA
```

```
9D00015C  00000000   NOP
33:
34:                  // Fonction DispMotif
35:                  // Compose un motif de n lignes
36:                   void DispMotif (uint8_t nbli, char symb1,
char symb2)
37:                  {
// ajout 8 mots de 32 bits dans stack
9D000160  27BDFFE0   ADDIU SP, SP, -32
9D000164  AFBF001C   SW RA, 28(SP)      // push RA
9D000168  AFBE0018   SW S8, 24(SP)      // push S8
9D00016C  03A0F021   ADDU S8, SP, ZERO  // push S8 = SP
9D000170  00A01821   ADDU V1, A1, ZERO
9D000174  00C01021   ADDU V0, A2, ZERO
9D000178  A3C40020   SB A0, 32(S8)      // val P1 => nbli
9D00017C  A3C30024   SB V1, 36(S8)      // val P2 => symb1
9D000180  A3C20028   SB V0, 40(S8)      // val P3 => symb2
38:                     uint8_t i;
39:                     for (i=1; i <= nbli; i++) {
9D000184  24020001   ADDIU V0, ZERO, 1
9D000188  A3C20010   SB V0, 16(S8)
9D00018C  0B400074   J 0x9D0001D0
9D000190  00000000   NOP
9D0001C4  93C20010   LBU V0, 16(S8)
9D0001C8  24420001   ADDIU V0, V0, 1
9D0001CC  A3C20010   SB V0, 16(S8)
9D0001D0  93C30010   LBU V1, 16(S8)
9D0001D4  93C20020   LBU V0, 32(S8)
9D0001D8  0043102B   SLTU V0, V0, V1
9D0001DC  1040FFED   BEQ V0, ZERO, 0x9D000194
9D0001E0  00000000   NOP
40:                    ComposeLigne(i, symb1, nbli-i, symb2);
9D000194  93C40010   LBU A0, 16(S8)          // A0 (P1) = i
9D000198  83C50024   LB A1, 36(S8)           // A1 (P2) = symb1
9D00019C  93C30020   LBU V1, 32(S8)          // V1 = nbli
9D0001A0  93C20010   LBU V0, 16(S8)          // V0 = i
9D0001A4  00621023   SUBU V0, V1, V0
9D0001A8  304200FF   ANDI V0, V0, 255
9D0001AC  00401821   ADDU V1, V0, ZERO       // V1 = nbli - i
9D0001B0  83C20028   LB V0, 40(S8)
9D0001B4  00603021   ADDU A2, V1, ZERO    // A2 (P3) = nbli-i
9D0001B8  00403821   ADDU A3, V0, ZERO       // A3 (P4) = symb2
9D0001BC  0F400038   JAL ComposeLigne
9D0001C0  00000000   NOP
41:                      }
42:                  }    // retour de DispMotif
9D0001E4  03C0E821   ADDU SP, S8, ZERO       // SP = S8
9D0001E8  8FBF001C   LW RA, 28(SP)           // pop RA
9D0001EC  8FBE0018   LW S8, 24(SP)           // pop S8
9D0001F0  27BD0020   ADDIU SP, SP, 32        // maj SP
9D0001F4  03E00008   JR RA                   // retour
```

```
9D0001F8  00000000   NOP
43:
44:
45:                  uint16_t ExtractHeures(uint32_t nbSecTot)
46:                  {
9D0001FC  27BDFFF8   ADDIU SP, SP, -8        // ajout 2 mots
9D000200  AFBE0004   SW S8, 4(SP)            // push S8
9D000204  03A0F021   ADDU S8, SP, ZERO       // S8 = SP
9D000208  AFC40008   SW A0, 8(S8)       // A0 (P1) => nbsecTot
47:                      return (nbSecTot / 3600);
9D00020C  8FC30008   LW V1, 8(S8)
9D000210  24020E10   ADDIU V0, ZERO, 3600
9D000214  0062001B   DIVU V1, V0
9D000218  004001F4   TEQ V0, ZERO
9D00021C  00001810   MFHI V1, 0
9D000220  00001012   MFLO V0, 0
9D000224  3042FFFF   ANDI V0, V0, -1    // résultat dans V0
48:                  }
// mécanisme de retour de ExtractHeures
9D000228  03C0E821   ADDU SP, S8, ZERO       // SP = S8
9D00022C  8FBE0004   LW S8, 4(SP)            // pop S8
9D000230  27BD0008   ADDIU SP, SP, 8         // maj SP
9D000234  03E00008   JR RA                   // retour
9D000238  00000000   NOP
49:
50:                            uint16_t  ExtractMinutes(uint32_t
nbSecTot, uint16_t NbH)
51:                  {
9D00023C  27BDFFF8   ADDIU SP, SP, -8        // ajout 2 mots
9D000240  AFBE0004   SW S8, 4(SP)            // push S8
9D000244  03A0F021   ADDU S8, SP, ZERO       // S8 = SP
9D000248  AFC40008   SW A0, 8(S8)       // A0 (P1) => nbsecTot
9D00024C  00A01021   ADDU V0, A1, ZERO
9D000250  A7C2000C   SH V0, 12(S8)      // A1 (P2) => NbH
52:                 return ( (nbSecTot - ( NbH * 3600)) / 60);
9D000254  97C2000C   LHU V0, 12(S8)
9D000258  00401821   ADDU V1, V0, ZERO
9D00025C  00031100   SLL V0, V1, 4
9D000260  00401821   ADDU V1, V0, ZERO
9D000264  00031100   SLL V0, V1, 4
9D000268  00431023   SUBU V0, V0, V1
9D00026C  00021900   SLL V1, V0, 4
9D000270  00621023   SUBU V0, V1, V0
9D000274  00021023   SUBU V0, ZERO, V0
9D000278  00401821   ADDU V1, V0, ZERO
9D00027C  8FC20008   LW V0, 8(S8)
9D000280  00621821   ADDU V1, V1, V0
9D000284  2402003C   ADDIU V0, ZERO, 60
9D000288  0062001B   DIVU V1, V0
9D00028C  004001F4   TEQ V0, ZERO
9D000290  00001810   MFHI V1, 0
```

```
9D000294  00001012   MFLO V0, 0
9D000298  3042FFFF   ANDI V0, V0, -1
53:                  }
// mécanisme de retour de ExtractMinutes
9D00029C  03C0E821   ADDU SP, S8, ZERO      // SP = S8
9D0002A0  8FBE0004   LW S8, 4(SP)           // pop S8
9D0002A4  27BD0008   ADDIU SP, SP, 8        // maj SP
9D0002A8  03E00008   JR RA                  // retour
9D0002AC  00000000   NOP
54:
55:              uint16_t ExtractSecondes(uint32_t nbSecTot,
                                          uint16_t NbH, uint16_t NbM)
56:                  {
9D0002B0  27BDFFF8   ADDIU SP, SP, -8       // ajout 2 mots
9D0002B4  AFBE0004   SW S8, 4(SP)           // push S8
9D0002B8  03A0F021   ADDU S8, SP, ZERO      // S8 = SP
9D0002BC  AFC40008   SW A0, 8(S8)       // A0 (P1) => nbsecTot
9D0002C0  00A01821   ADDU V1, A1, ZERO
9D0002C4  00C01021   ADDU V0, A2, ZERO
9D0002C8  A7C3000C   SH V1, 12(S8)      // A1 (P2) => NbH
9D0002CC  A7C20010   SH V0, 16(S8)      // A2 (P3) => NbM
57:          return ( nbSecTot - ( NbH * 3600) - (NbM *60) );
9D0002D0  97C2000C   LHU V0, 12(S8)
9D0002D4  00021100   SLL V0, V0, 4
9D0002D8  00021900   SLL V1, V0, 4
9D0002DC  00621023   SUBU V0, V1, V0
9D0002E0  00401821   ADDU V1, V0, ZERO
9D0002E4  00031900   SLL V1, V1, 4
9D0002E8  00621023   SUBU V0, V1, V0
9D0002EC  00021023   SUBU V0, ZERO, V0
9D0002F0  3043FFFF   ANDI V1, V0, -1
9D0002F4  97C20010   LHU V0, 16(S8)
9D0002F8  00021080   SLL V0, V0, 2
9D0002FC  00022100   SLL A0, V0, 4
9D000300  00821023   SUBU V0, A0, V0
9D000304  00021023   SUBU V0, ZERO, V0
9D000308  3042FFFF   ANDI V0, V0, -1
9D00030C  00621021   ADDU V0, V1, V0
9D000310  3043FFFF   ANDI V1, V0, -1
9D000314  8FC20008   LW V0, 8(S8)
9D000318  3042FFFF   ANDI V0, V0, -1
9D00031C  00621021   ADDU V0, V1, V0
9D000320  3042FFFF   ANDI V0, V0, -1
58:                  }
// mécanisme de retour de ExtractSecondes
9D000324  03C0E821   ADDU SP, S8, ZERO      // SP = S8
9D000328  8FBE0004   LW S8, 4(SP)           // pop S8
9D00032C  27BD0008   ADDIU SP, SP, 8        // maj SP
9D000330  03E00008   JR RA                  // retour
9D000334  00000000   NOP
59:
```

```
60:
61:                     typedef struct {
62:                         uint16_t NbH;
63:                         uint16_t NbM;
64:                         uint16_t NbS;
65:                     } S_HMS;
66:
67:                     S_HMS ExtractHMS(uint32_t nbSecTot)
68:                     {
9D000338  27BDFFF0    ADDIU SP, SP, -16       // ajout 4 mots
9D00033C  AFBE000C    SW S8, 12(SP)           // push S8
9D000340  03A0F021    ADDU S8, SP, ZERO       // S8 = SP
9D000344  00801021    ADDU V0, A0, ZERO
9D000348  AFC50014    SW A1, 20(S8)       // A1 => nbSecTot
69:                         S_HMS tmp;
70:
71:                         tmp.NbH = nbSecTot / 3600;
9D00034C  8FC40014    LW A0, 20(S8)
9D000350  24030E10    ADDIU V1, ZERO, 3600
9D000354  0083001B    DIVU A0, V1
9D000358  006001F4    TEQ V1, ZERO
9D00035C  00002010    MFHI A0, 0
9D000360  00001812    MFLO V1, 0
9D000364  3063FFFF    ANDI V1, V1, -1
9D000368  A7C30000    SH V1, 0(S8)        // tmp.NbH 0> 0(S8)
72:             tmp.NbM = (nbSecTot - ( tmp.NbH * 3600)) / 60;
9D00036C  97C30000    LHU V1, 0(S8)
9D000370  00602021    ADDU A0, V1, ZERO
9D000374  00041900    SLL V1, A0, 4
9D000378  00602021    ADDU A0, V1, ZERO
9D00037C  00041900    SLL V1, A0, 4
9D000380  00641823    SUBU V1, V1, A0
9D000384  00032100    SLL A0, V1, 4
9D000388  00831823    SUBU V1, A0, V1
9D00038C  00031823    SUBU V1, ZERO, V1
9D000390  00602021    ADDU A0, V1, ZERO
9D000394  8FC30014    LW V1, 20(S8)
9D000398  00832021    ADDU A0, A0, V1
9D00039C  2403003C    ADDIU V1, ZERO, 60
9D0003A0  0083001B    DIVU A0, V1
9D0003A4  006001F4    TEQ V1, ZERO
9D0003A8  00002010    MFHI A0, 0
9D0003AC  00001812    MFLO V1, 0
9D0003B0  3063FFFF    ANDI V1, V1, -1
9D0003B4  A7C30002    SH V1, 2(S8)        // tmp.NbM => 2(S8)
73:                             tmp.NbS = nbSecTot - ( tmp.NbH *
3600) - (tmp.NbM *60);
9D0003B8  97C30000    LHU V1, 0(S8)
9D0003BC  00031900    SLL V1, V1, 4
9D0003C0  00032100    SLL A0, V1, 4
9D0003C4  00831823    SUBU V1, A0, V1
```

```
9D0003C8  00602021   ADDU A0, V1, ZERO
9D0003CC  00042100   SLL A0, A0, 4
9D0003D0  00831823   SUBU V1, A0, V1
9D0003D4  00031823   SUBU V1, ZERO, V1
9D0003D8  3064FFFF   ANDI A0, V1, -1
9D0003DC  97C30002   LHU V1, 2(S8)
9D0003E0  00031880   SLL V1, V1, 2
9D0003E4  00032900   SLL A1, V1, 4
9D0003E8  00A31823   SUBU V1, A1, V1
9D0003EC  00031823   SUBU V1, ZERO, V1
9D0003F0  3063FFFF   ANDI V1, V1, -1
9D0003F4  00831821   ADDU V1, A0, V1
9D0003F8  3064FFFF   ANDI A0, V1, -1
9D0003FC  8FC30014   LW V1, 20(S8)
9D000400  3063FFFF   ANDI V1, V1, -1
9D000404  00831821   ADDU V1, A0, V1
9D000408  3063FFFF   ANDI V1, V1, -1
9D00040C  A7C30004   SH V1, 4(S8)  // tmp.NbS => 4(S8)
74:                      return ( tmp );
9D000410  8FC30000   LW V1, 0(S8)
9D000414  A8430003   SWL V1, 3(V0)
9D000418  B8430000   SWR V1, 0(V0)
9D00041C  97C30004   LHU V1, 4(S8)
9D000420  A4430004   SH V1, 4(V0)
75:                  }
// mécanisme de retour de ExtractHMS
9D000424  03C0E821   ADDU SP, S8, ZERO  // SP = S8
9D000428  8FBE000C   LW S8, 12(SP)       // pop S8
9D00042C  27BD0010   ADDIU SP, SP, 16    // maj SP
9D000430  03E00008   JR RA               // retour
9D000434  00000000   NOP
76:
77:                  void main() {
9D000438  27BDFFD0   ADDIU SP, SP, -48  // ajout 12 mots
9D00043C  AFBF002C   SW RA, 44(SP)       // push RA
9D000440  AFBE0028   SW S8, 40(SP)       // push S8
9D000444  03A0F021   ADDU S8, SP, ZERO  // S8 = SP
78:
79:                     uint32_t NbSecT = 3675;
9D000448  24020E5B   ADDIU V0, ZERO, 3675
9D00044C  AFC20010   SW V0, 16(S8)
80:                     uint16_t H, M, S;
81:                     S_HMS ResHMS;
82:                     bool match;
83:
84:                     // Appel des fonctions d'extractions
85:                     H = ExtractHeures(NbSecT);
9D000450  8FC40010   LW A0, 16(S8)
9D000454  0F40007F   JAL ExtractHeures
9D000458  00000000   NOP
9D00045C  A7C20014   SH V0, 20(S8)
```

```
86:                        M = ExtractMinutes(NbSecT, H);
9D000460  97C20014    LHU V0, 20(S8)
9D000464  8FC40010    LW A0, 16(S8)
9D000468  00402821    ADDU A1, V0, ZERO
9D00046C  0F40008F    JAL ExtractMinutes
9D000470  00000000    NOP
9D000474  A7C20016    SH V0, 22(S8)
87:                        S = ExtractSecondes(NbSecT, H, M);
9D000478  97C30014    LHU V1, 20(S8)
9D00047C  97C20016    LHU V0, 22(S8)
9D000480  8FC40010    LW A0, 16(S8)
9D000484  00602821    ADDU A1, V1, ZERO
9D000488  00403021    ADDU A2, V0, ZERO
9D00048C  0F4000AC    JAL ExtractSecondes
9D000490  00000000    NOP
9D000494  A7C20018    SH V0, 24(S8)
88:
89:                        ResHMS = ExtractHMS(NbSecT);
9D000498  27C2001C    ADDIU V0, S8, 28
9D00049C  00402021    ADDU A0, V0, ZERO
9D0004A0  8FC50010    LW A1, 16(S8)
9D0004A4  0F4000CE    JAL ExtractHMS
9D0004A8  00000000    NOP
90:
91:                        if (H = ResHMS.NbH) match = true;
9D0004AC  97C2001C    LHU V0, 28(S8)
9D0004B0  A7C20014    SH V0, 20(S8)
9D0004B4  97C20014    LHU V0, 20(S8)
9D0004B8  10400003    BEQ V0, ZERO, 0x9D0004C8
9D0004BC  00000000    NOP
9D0004C0  24020001    ADDIU V0, ZERO, 1
9D0004C4  A3C2001A    SB V0, 26(S8)
92:                        if (M = ResHMS.NbM) match = true;
9D0004C8  97C2001E    LHU V0, 30(S8)
9D0004CC  A7C20016    SH V0, 22(S8)
9D0004D0  97C20016    LHU V0, 22(S8)
9D0004D4  10400003    BEQ V0, ZERO, 0x9D0004E4
9D0004D8  00000000    NOP
9D0004DC  24020001    ADDIU V0, ZERO, 1
9D0004E0  A3C2001A    SB V0, 26(S8)
93:                        if (S = ResHMS.NbS) match = true;
9D0004E4  97C20020    LHU V0, 32(S8)
9D0004E8  A7C20018    SH V0, 24(S8)
9D0004EC  97C20018    LHU V0, 24(S8)
9D0004F0  10400003    BEQ V0, ZERO, 0x9D000500
9D0004F4  00000000    NOP
9D0004F8  24020001    ADDIU V0, ZERO, 1
9D0004FC  A3C2001A    SB V0, 26(S8)
94:
95:
```

```
96:                     // Appel de la fonction  DispMotif
97:                     DispMotif(5, '*', '+');
9D000500  24040005   ADDIU A0, ZERO, 5
9D000504  2405002A   ADDIU A1, ZERO, 42
9D000508  2406002B   ADDIU A2, ZERO, 43
9D00050C  0F400058   JAL DispMotif
9D000510  00000000   NOP
98:                     DispMotif(7, '!', '=');
9D000514  24040007   ADDIU A0, ZERO, 7
9D000518  24050021   ADDIU A1, ZERO, 33
9D00051C  2406003D   ADDIU A2, ZERO, 61
9D000520  0F400058   JAL DispMotif
9D000524  00000000   NOP
99:                     }
9D000528  03C0E821   ADDU SP, S8, ZERO
9D00052C  8FBF002C   LW RA, 44(SP)
9D000530  8FBE0028   LW S8, 40(SP)
9D000534  27BD0030   ADDIU SP, SP, 48
9D000538  03E00008   JR RA
9D00053C  00000000   NOP
```

Illustration graphique de la situation lors de l'appel de la fonction ExtractHeures (situation après exécution du prologue de la fonction) :

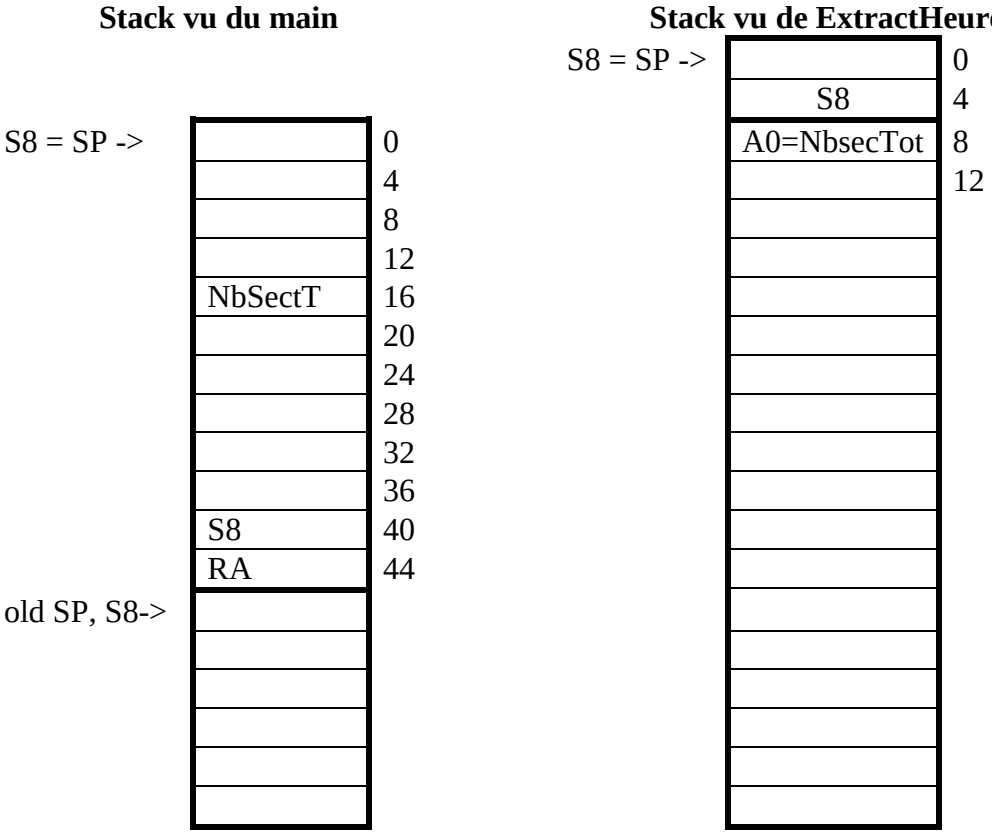


Illustration graphique de la situation lors de l'appel de la fonction ExtractMinutes (situation après exécution du prologue de la fonction) :

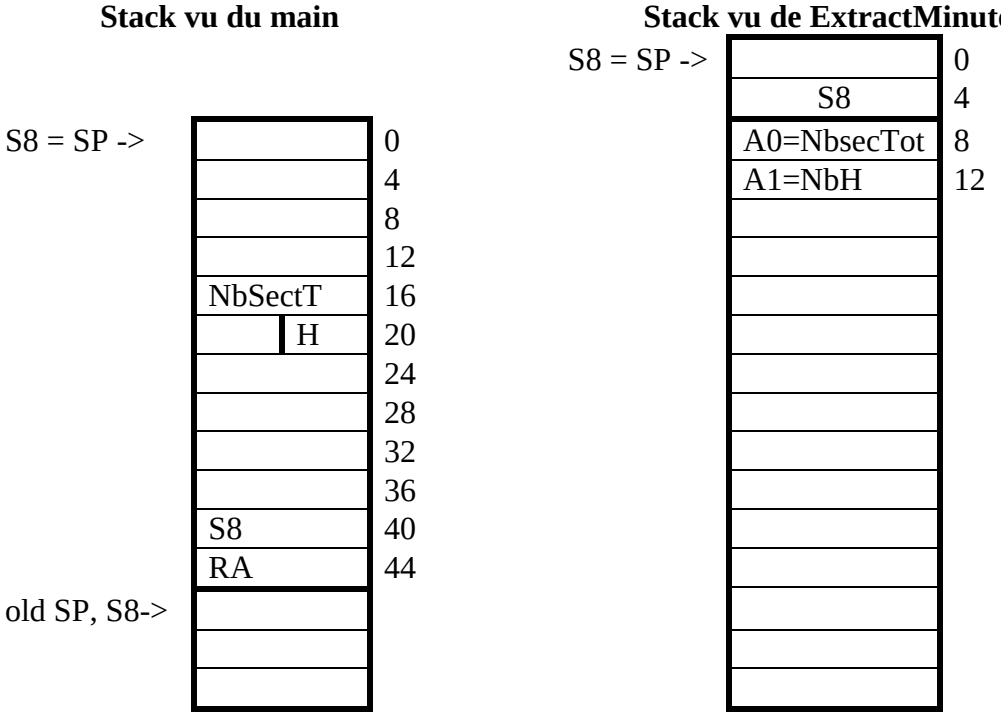


Illustration graphique de la situation lors de l'appel de la fonction ExtractSecondes (situation après exécution du prologue de la fonction) :

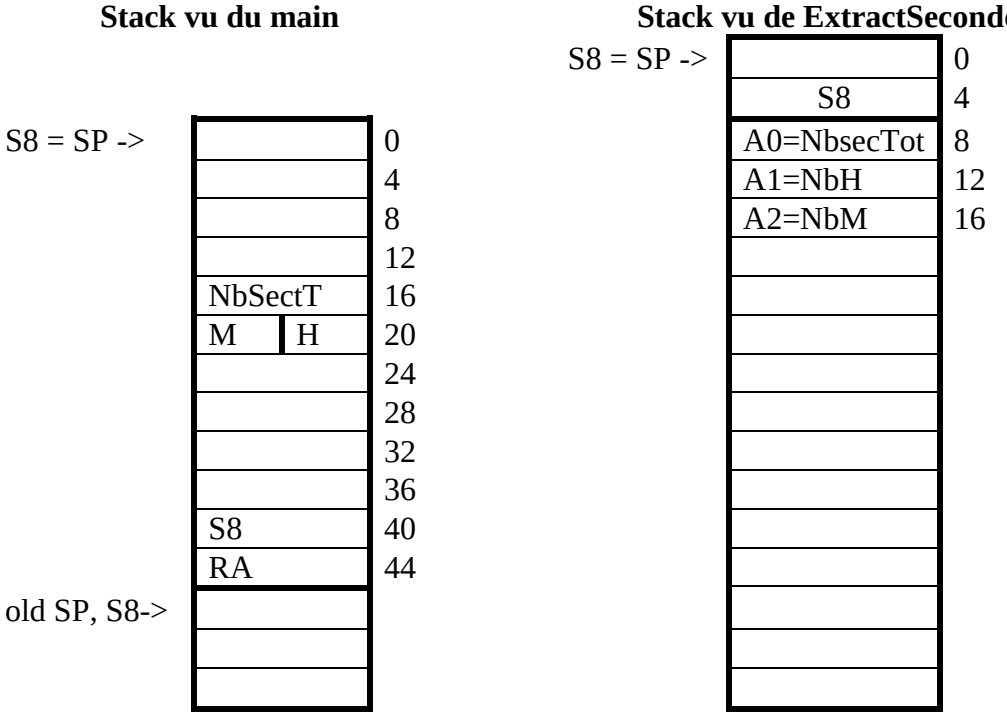
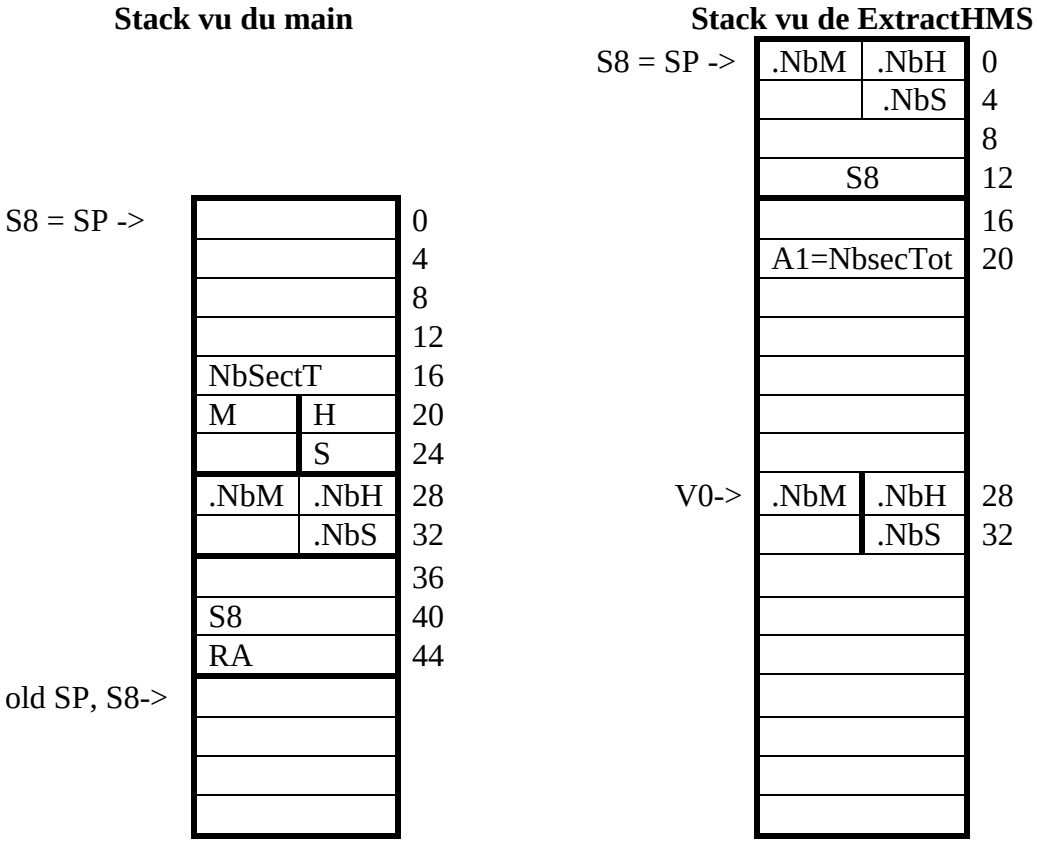


Illustration graphique de la situation lors de l'appel de la fonction **ExtractHMS**. (Situation juste avant exécution de l'épilogue de la fonction) :

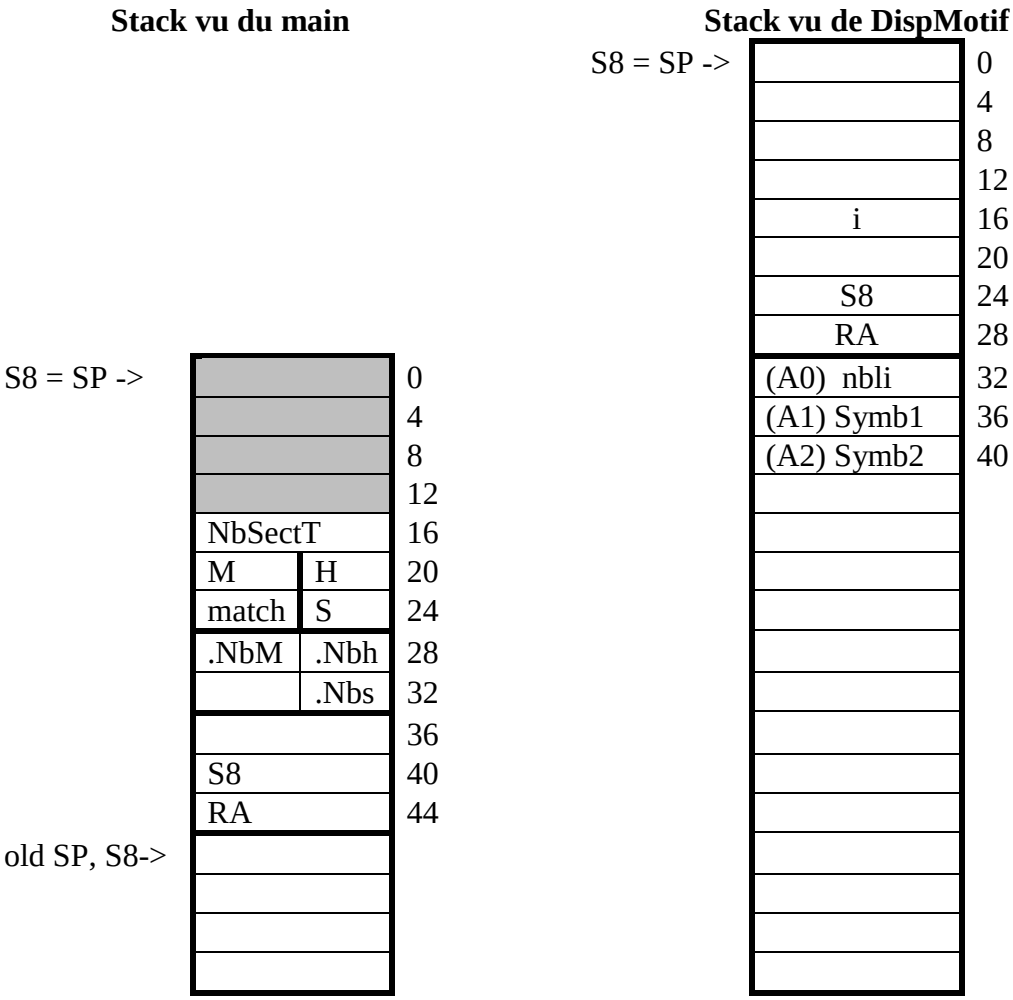


Quelques remarques :

- On peut observer que les variables H, M, S utilisées comme retour et fournie en paramètres sont placée dans le stack du main.
- On peut aussi constater que les fonctions utilisant une zone du stack qui correspond au début du stack vu du main.

PARTIE B)

Complétez la situation du stack dans la fonction **DispMotif** juste avant d'effectuer le **JAL ComposeLigne**.



Lors de l'appel de la fonction DispMotif, les valeurs des paramètres sont placées dans A0, A1 et A2.

```
97:                         DispMotif(5, '*', '+');
9D000500  24040005   ADDIU A0, ZERO, 5
9D000504  2405002A   ADDIU A1, ZERO, 42
9D000508  2406002B   ADDIU A2, ZERO, 43
```

Dans le prologue de la fonction DispMotif A0, A1 et A2 sont copiés en 32(S8), 36(S8) et 40(S8), ce qui correspond à l'emplacement de nbli, Symb1 et Symb2.

Le i de la boucle for est situé en 16(S8).