

# **Correction 19-20 s2 |**

# **Compilation**

2025-12-30

**F. B**  
Universite de Montpellier  
2025  
*M1*

## Contents

I.	Exercice 1 .....	3
I.1.	Que fait f .....	3
I.2.	PP .....	3
I.3.	UPP .....	3
I.4.	RTL .....	3
I.5.	ERTL .....	4
II.	Exercice 2 .....	5
II.1.	.....	5
II.2.	.....	6
II.3.	.....	6
II.4.	.....	7
III.	Exercice 3 .....	8
III.1.	.....	8
III.2.	.....	8

## I. Exercice 1

### I.1. Que fait f

$$\begin{aligned}
 f(3) &= 2 \times 3 - f(2) - 1 \\
 &= 5 - f(2) \\
 &= 5 - (2 \times 2 - f(1) - 1) - 1 \\
 &= 5 - (4 - (2 \times 1 - f(0) - 1) - 1) - 1 \\
 &= 5 - (4 - 1 - 1 - 1) - 1 = 5 - 1 - 1 = 3
 \end{aligned}$$

$$f(5) = 5$$

Elle rend elle même, si  $f(4)$  on aura 4

### I.2. PP

```

1 function f(n : integer) : integer :
2   if n = 0
3     return 0
4   else
5     return (2 * n) - f(n-1) - 1

```

### I.3. UPP

```

1 function f(n) :
2   if n = 0
3     return 0
4   else
5     return (2 * n) - f(n-1) - 1

```

### I.4. RTL

```

1 function f(%0) : %1 :
2   var %0 %1 %2 %3 %4
3   entry start
4   exit final
5
6   start : beqz %0 -> base,rec
7   base : li %1 , 0 -> final
8
9   rec    : sub %2, %0, 1 -> call      ; on calcule n-1
10  call   : call %3, f(%2) -> mult      ; On appelle f(n-1)
11
12  mult   : mul %4, %0, 2 -> sub1      ; %4 <- 2 * n
13  sub1   : sub %4, %4, %3 -> sub2      ; %4 <- (2*n) - f(n-1)
14  sub2   : sub %1, %4, 1 -> final      ; %1 <- Tout ça - 1
15  final  :

```

## I.5. ERTL

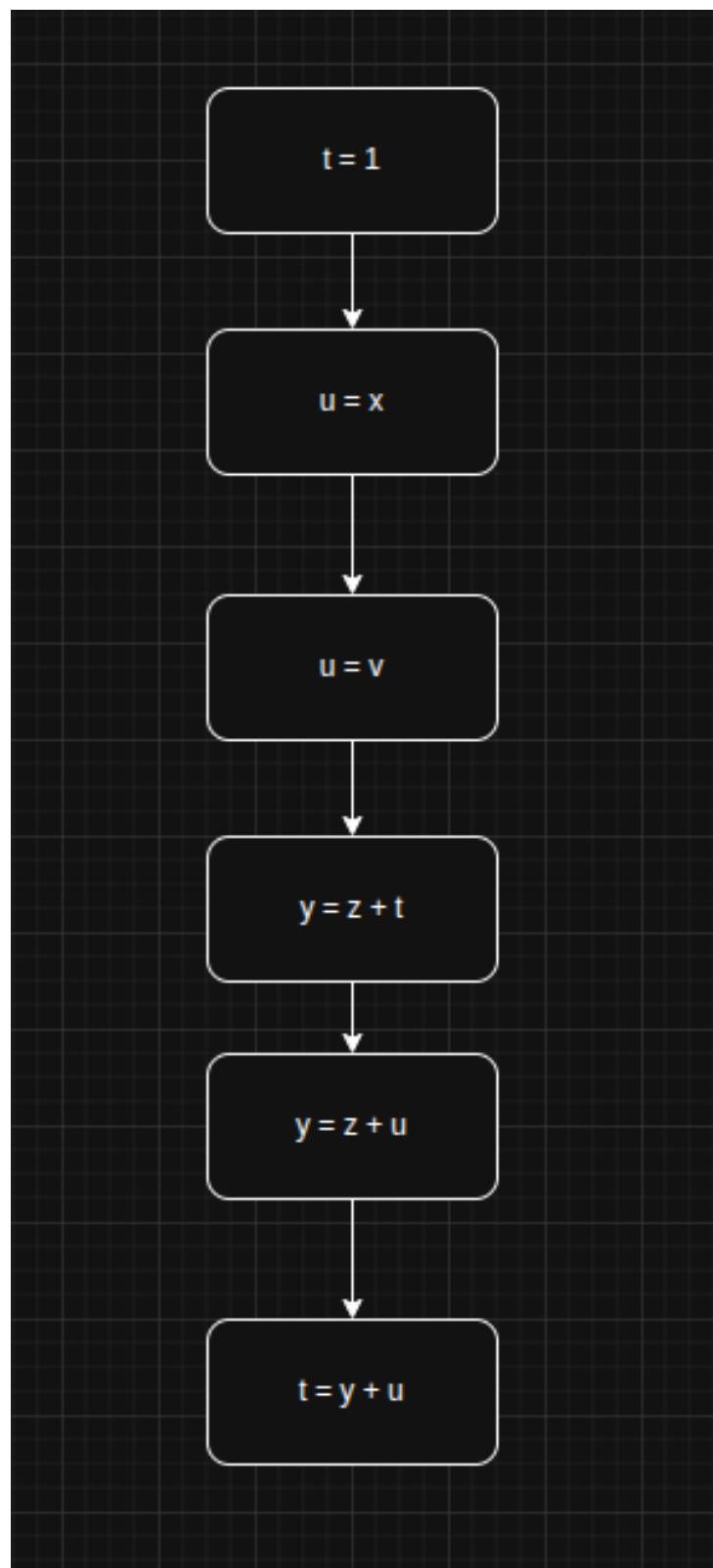
```

1  procedure f(1)
2  var %0 %1 %2 %3 %4 %save_ra %save_s0
3  entry f_prologue
4  exit f_end
5
6  ; copy paste du cheat sheet
7  f_entry : newframe -> f_save
8  f_save  : move %save_ra $ra -> f_save_s
9  f_save_s: move %save_s0 $s0 -> f_arg
10 f_arg   : move %s0 $a0 -> f_body
11
12 f_body : beqz %0 -> base,rec
13 base : li %1, 0 -> f_return
14
15 rec : sub %2,%s0,1 -> prep_call      ; utilise s0 car appel rec
16 prep_call : move %a0, %2 -> call    ; pr faire appel, on met dans a0
17 call : call f(1) -> get_res
18 get_res  : move %3 $v0 -> mult
19
20 mult : mul %4, %s0, 2 -> sub1      ; utilise s0
21 sub1 : sub %4, %4, %3 -> sub2
22 sub2 : sub %1, %4, 1 -> f_return
23
24 ; copy paste du cheat sheet
25 f_return:  move $v0 %1 -> f_rest_s
26 f_rest_s: move $s0 %save_s0 -> f_rest_ra;
27 f_rest_ra: move $ra %save_ra -> f_del
28 f_del:    delframe -> f_jr
29 f_jr:     jr $ra -> f_end
30
31 f_end:

```

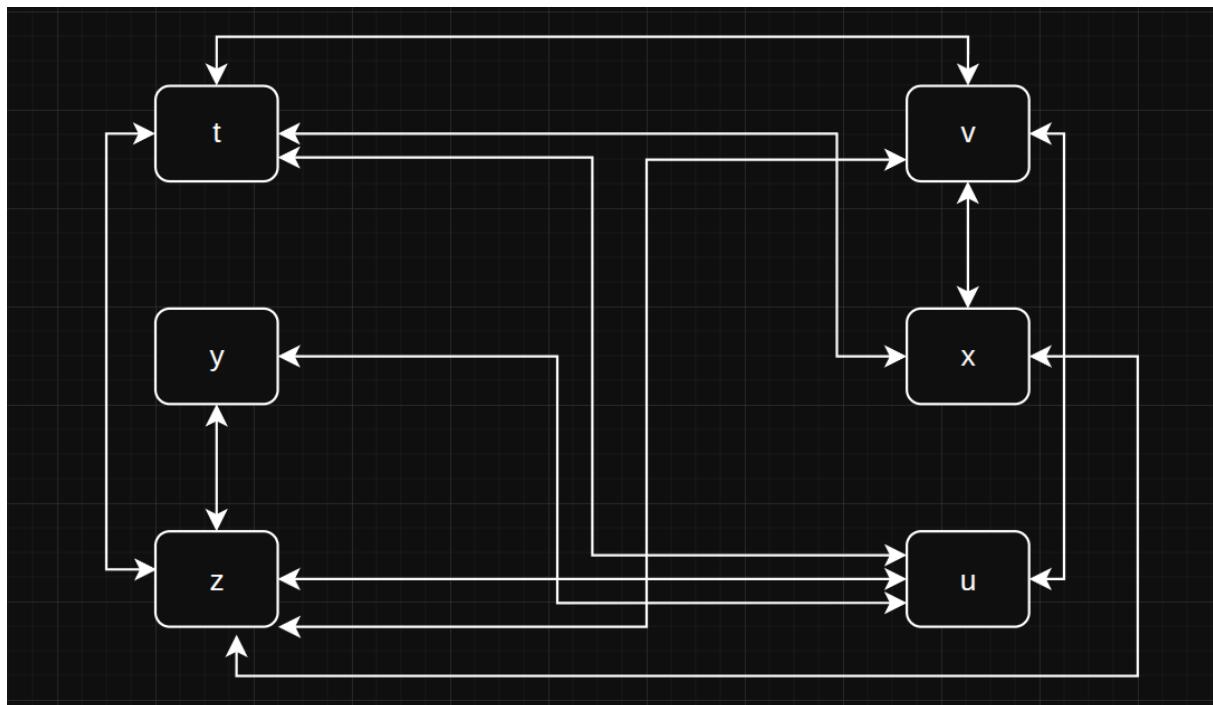
## II. Exercice 2

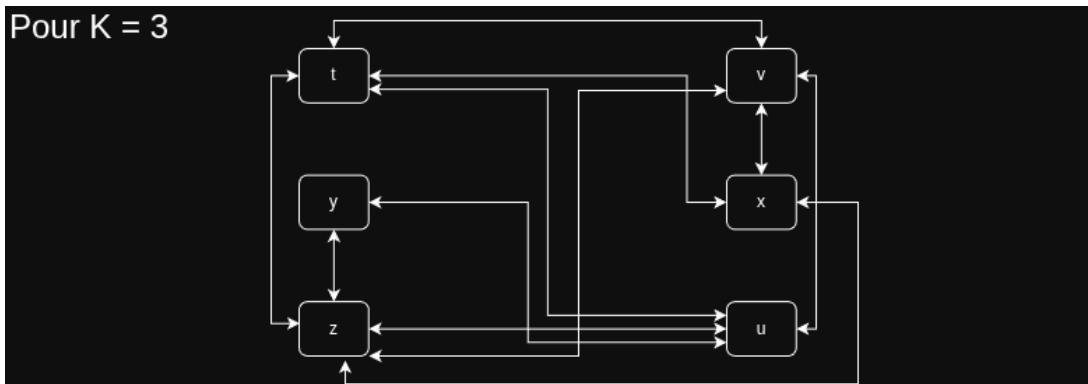
### II.1.



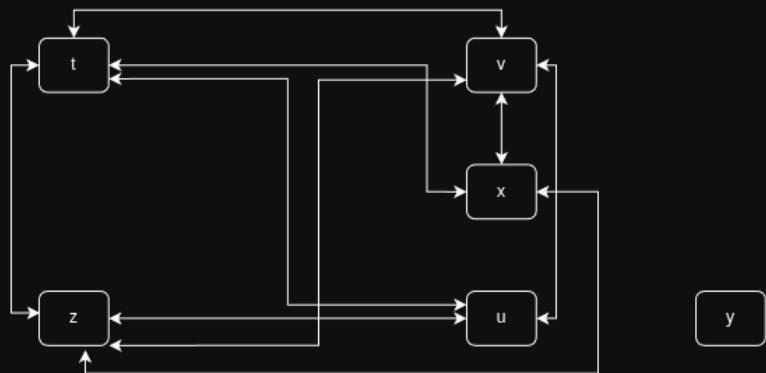
**II.2.**

```
1 {x,v,z}
2 t = 1 {z,v,x}
3 u = x {z,t,v,x}
4 u = v {z,t,v}
5 y = z + t {z,t,u}
6 y = z + u {z,u}
7 t = y + u {y,u}
8 {t,u}
```

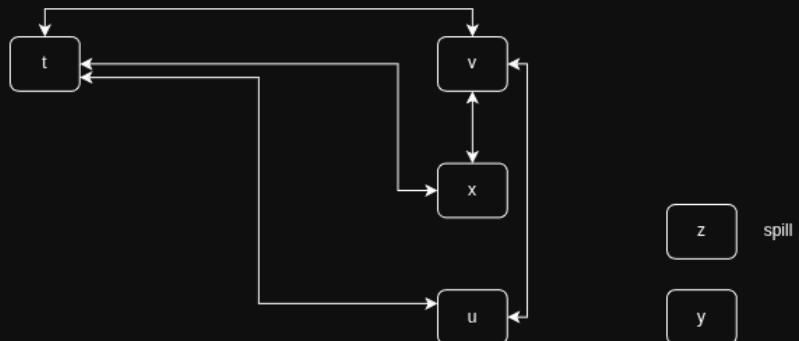
**II.3.**

**II.4.**

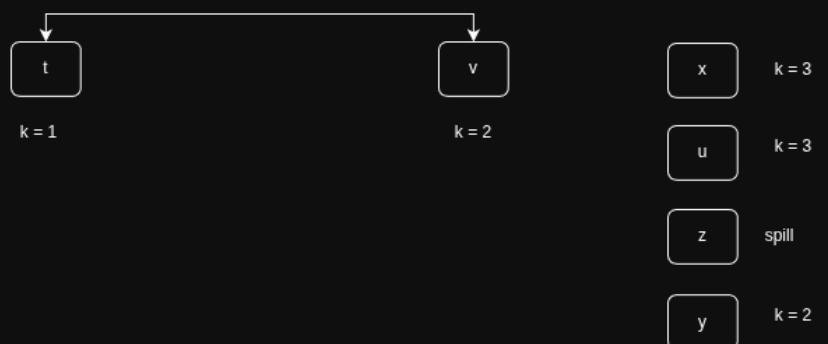
on cherche un sommet qui a  $< 3$  de noeud, on a y



comme on a rien d'autre avec noeud  $< 3$ , on spill, je choisi z



On enlève u et x car noeud  $< 3$ , et on colorie en remontant



flemme de faire  $k = 2$  dsl

### III. Exercice 3

#### III.1.

Les états d'un automate peuvent-être traduis au sein de la VM par des labels. Chaque état aura donc son étiquette et il sera possible de sauter à un label pour effectuer une transition. Pour chaque état nous chargerons le caractère correspondant

#### III.2.

```

1  (vm-automate '(LISP
2    (LABEL start)
3    (JMP etat0)           ; L'état initial est 0
4
5    ;; --- ÉTAT 0 (Non Final) ---
6    (LABEL etat0)
7    (MOVE (:CONST etat0) R2) ; (Optionnel) Pour suivre l'état courant dans
8    R2
9    (BNULL R0 refuser)    ; Si la liste R0 est vide, on rejette (0 n'est pas
10   final)
11   (CAR R0 R1)          ; R1 prend la valeur du caractère courant (tête de
12   liste)
13   (CDR R0 R0)          ; R0 avance au caractère suivant (queue de liste)
14   (CMP R1 (:CONST b)) ; Compare le caractère lu avec 'b'
15   (JEQ etat1)          ; Si égal, transition vers l'état 1
16   (CMP R1 (:CONST a)) ; Compare le caractère lu avec 'a'
17   (JEQ etat3)          ; Si égal, transition vers l'état 3
18   (JMP refuser)        ; Si c'est autre chose transition invalide -> rejet
19
20   ;; --- ÉTAT 1 (Non Final) ---
21   (LABEL etat1)
22   (MOVE (:CONST etat1) R2)
23   (BNULL R0 refuser)    ; Si vide, on rejette (1 n'est pas final)
24   (CAR R0 R1)
25   (CDR R0 R0)
26   (CMP R1 (:CONST b))
27   (JEQ etat1)          ; Boucle sur 'b'
28   (CMP R1 (:CONST a))
29   (JEQ etat3)
30   (JMP refuser)
31
32   ;; --- ÉTAT 2 (Non Final) ---
33   (LABEL etat2)
34   (MOVE (:CONST etat2) R2)
35   (BNULL R0 refuser)    ; Si vide, on rejette

```

```
33  (CAR R0 R1)
34  (CDR R0 R0)
35  (CMP R1 (:CONST a))
36  (JEQ etat0)
37  (CMP R1 (:CONST b))
38  (JEQ etat2)
39  (JMP refuser)
40
41  ;; --- ETAT 3 (Final) ---
42  (LABEL etat3)
43  (MOVE (:CONST etat3) R2)
44  (BNULL R0 accepter) ; Si la liste est vide ici (État final)
45  (CAR R0 R1)
46  (CDR R0 R0)
47  (CMP R1 (:CONST b))
48  (JEQ etat2)
49  (JMP refuser)
50
51  ;; --- BLOCS DE FIN ---
52  (LABEL accepter)
53  (MOVE R2 R0) ; On retourne le chemin
54  (HALT) ; Arrêt de la machine
55
56  (LABEL refuser)
57  (MOVE (:CONST nil) R0) ; On retourne Faux/Nil
58  (HALT)
59 ))
```