2018

1 Partie D

1.1 Pseudo Pascal

```
f(n : integer) : integer
    if n <= 1 then
        f := n
    else
        f:= f(n - 1) + f(n - 2)
```

1.2 UPP

```
f(n)
    if n <= 1 then
        f := n
    else
        f:= f(n - 1) + f(n - 2)
```

1.3 RTL

```
function f(%0): %1
var %0, %1, %2, %3, %4, %5, %6
entry f1
exit f0

f1: li %1, 0         -> f2
f2: slt %2, %0, 2    -> f3
f3: beq %2, 1        -> f5, f4
f4: addiu %3, %0, -1 -> f6
f6: call %4, f(%3)   -> f7
f7: addiu %5 %0, -2  -> f8
f8: call %6, f(%5)   -> f9
f9: addiu %1, %4, %6 -> f0
```

```
f5: li %1, %0        -> f0
```

1.4 ERTL

```
procedure f(1)
var %0, %1, %2, %3, %4, %5, %6, %7, %8, %9
entry f1

f1 : newframe            -> f2
f2 : move %7, $ra        -> f3
f3 : move %8, $s0        -> f4
f4 : move %9, $s1        -> f22
f22: move %0, $a0        -> f5
f5 : li %1, 0            -> f6
f6 : slt %2, %0, 2       -> f7
f7 : blez %2             -> f8, f22
f8 : addiu %3, %0, -1    -> f9
f9 : move $a0, %3        -> f10
f10: call(f1)            -> f11
f11: move $s0, $v0       -> f12
f12: addiu %5, %0, -2    -> f13
f13: move $a0, %5        -> f14
f14: call f(1)           -> f15
f15: move $s1, $v0       -> f16
f16: addiu %1, $s1, $s0  -> f0
f17: move $ra, %7        -> f18
f18: move $s0, %8        -> f19
f19: move $s1, %9        -> f20
f20: delframe            -> f21
f21: jr $ra              -> f22
f22: li %1, %0           -> f0
```

1.5 LTL

```
procedure f(1)
var 11
entry f1

f1 : newframe            -> f2
f2 : sets local(0), $ra  -> f3
f3 : sets local(4), $s0  -> f4
f4 : move $s0, $a0       -> f5
f5 : slt $a0, $a0, 2     -> f6
f6 : bleq $a0, 1         -> f7, f8
f8 : addiu $a0, $s0, -1  -> f9
```

```
f9 : call f              -> f11
f11: sets local(8) $s1   -> f12
f12: move $s1, $v0       -> f13
f13: addiu $a0, $s0, -2  -> f14
f14: call f              -> f15
f15: move $s0, $v0       -> f16
f16: addiu $v0, $s1, $s0 -> f0
f0 : j                   -> f17
f17: gets $ra, local(0)  -> f18
f18: gets $s0, local(4)  -> f19
f19: gets $s1, local(8)  -> f20
f20: delframe            -> f21
f21: jr $ra              -> f22
f7 : move $v0, $a0         -> f0
```

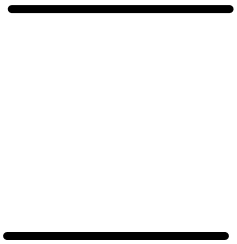
1.6 LIN

```
procedure f(1)
var 11
f1 :
newframe
sets local(0), $ra
sets local(4), $s0
move $s0, $a0
slt $a0, $a0, 2
bleq $a0, 1, f7
addiu $a0, $s0, -1
call f
sets local(8) $s1
move $s1, $v0
addiu $a0, $s0, -2
call f
move $s0, $v0
addiu $v0, $s1, $s0
f17:
gets $ra, local(0)
gets $s0, local(4)
gets $s1, local(8)
delframe
jr $ra
f7 :
move $v0, $a0
j f17
```

1.7 MIPS

```
main:   li $v0, 5
        syscall
        move $a0, $v0
        jal fib
        move $a0, $v0
        li $v0, 1
        syscall
        li $v0, 10
        syscall

fib:    addiu $sp, $sp, -12
        sw $ra, 8($sp)
        sw $s0, 4($sp)
        sw $s1, ($sp)
        move $s0, $a0
        li $t0, 1
        ble $s0, $t0, basis
        addiu $a0, $s0, -1
        jal fib
        move $s1, $v0
        addiu $a0, $s0, -2
        jal fib
        add $v0, $s1, $v0
        j exit
basis:  move $v0, $s0
exit:   lw $ra, 8($sp)
        lw $s0, 4($sp)
        lw $s1, ($sp)
        addiu $sp, $sp, 12
        jr $ra
```



```
terminale:

        addiu $a0, $a0, -1
        move $a1, $a2
        add $a2, $a2, $a1
        lw $ra, 0($sp)
        addiu $sp, 4
        j fib
```

1.8 Version récursive terminale en PP

```
f(n: integer, acc1 : integer, acc2 : integer) : integer
    if n <= 1 then
       f := acc1
    else
        f := f(n - 1, acc2, acc1 + acc2)

f(n, 0, 1)
```

1.9 Version récursive terminale en MIPS

```
addiu $a0, $a0, -1
move $a1, $a2
add $a2, $a2, $a1
lw $ra, 0($sp)
addiu $sp, 4
j fib
```

2 Partie L

Question 1

— HALT
— LABEL
— CMP
— MOVE
— JEQ

Question 2.1

Si on associe un label (une étiquette) à chaque état alors ces derniers seront liés à une adresse. Ainsi en utilisant l'instruction JMP on pourrait facilement passer d'un état à un autre. Ainsi une pile n'est pas nécessaire.

Question 2.2

```
(LABEL E0)
(CAR R0 R1)
(CDR R0 R0)
(CMP R1 a)
(JEQ E1)
(CMP R1 b)
(JEQ E0)
(MOVE NIL R0)
(HALT)

(LABEL E1)
(MOVE 'E1 R2)
(cmp R0)
(bnull H)
(CAR R0 R1)
(CDR R0 R0)
(CMP R1 a)
(JEQ E1)
(CMP R1 b)
(JEQ E0)
(CMP R1 c)
(JEQ E2)
(MOVE NIL R0)
(HALT)

(LABEL E2)
(MOVE 'E2 R2)
(cmp R0)
(bnull H)
```

```
(CAR R0 R1)
(CDR R0 R0)
(MOVE NIL R0)
(HALT)

(LABEL H)
(MOVE R2 R0)
(HALT)
```

Question 3