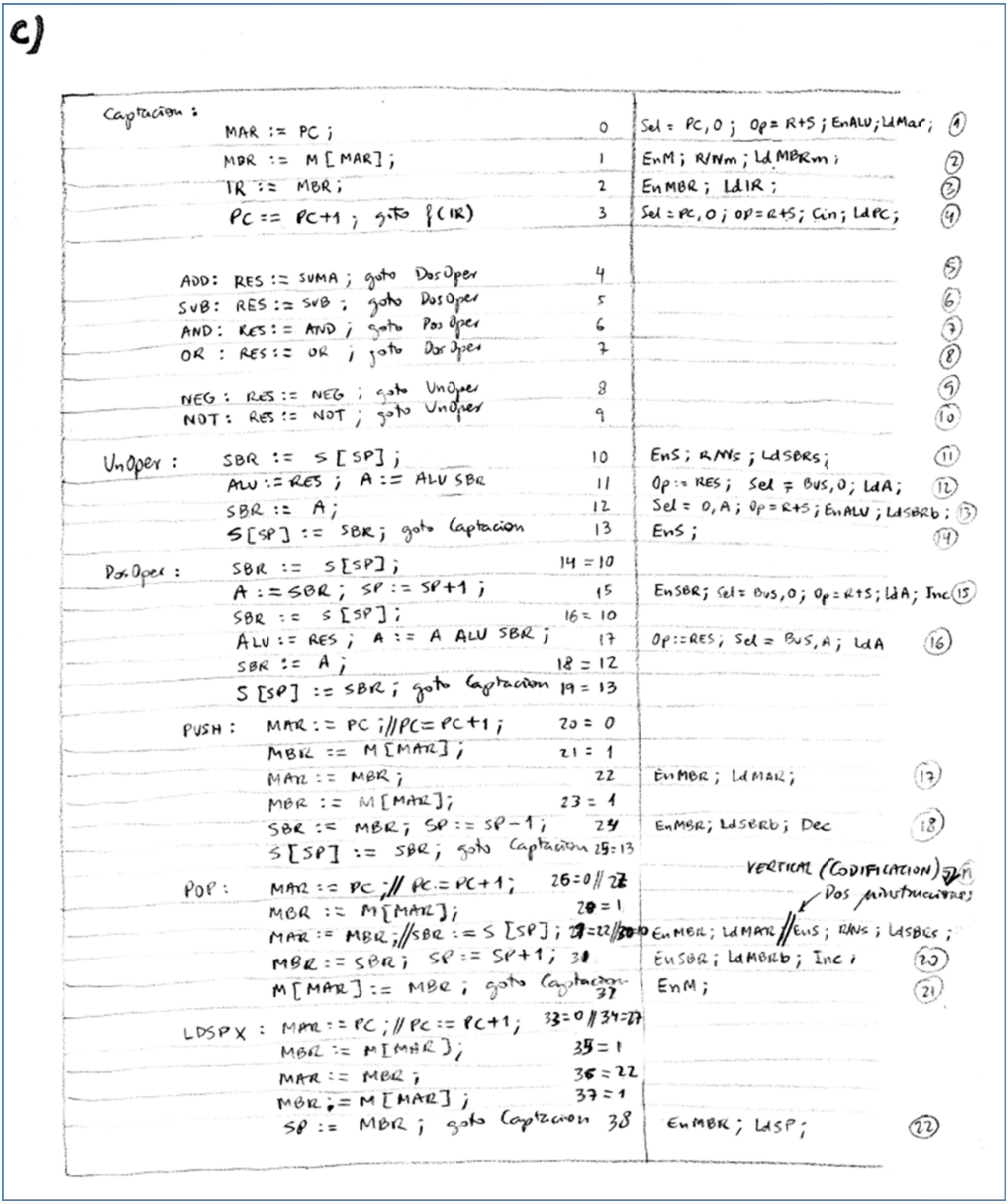
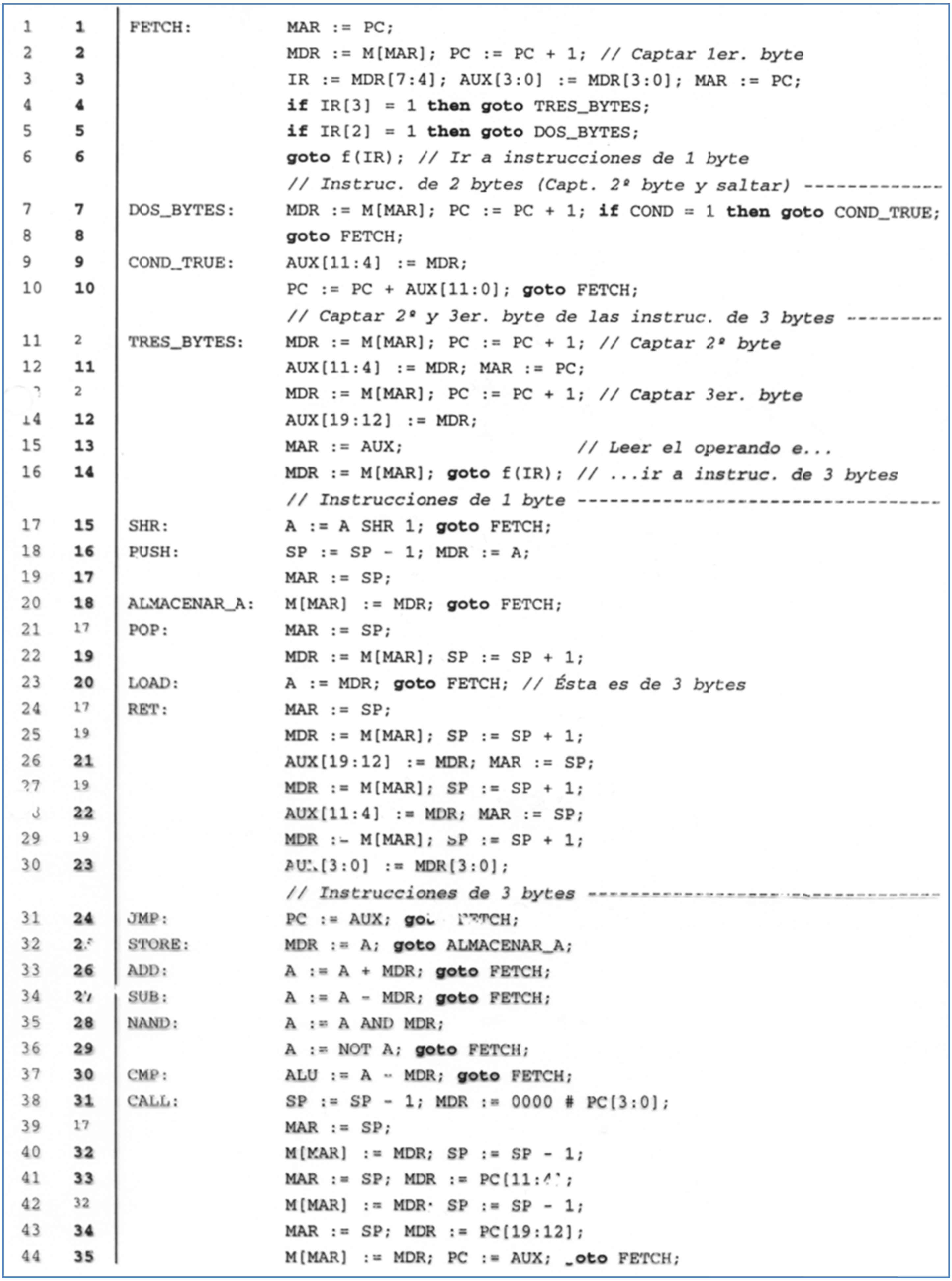
## Problema UC 3.c)



c)

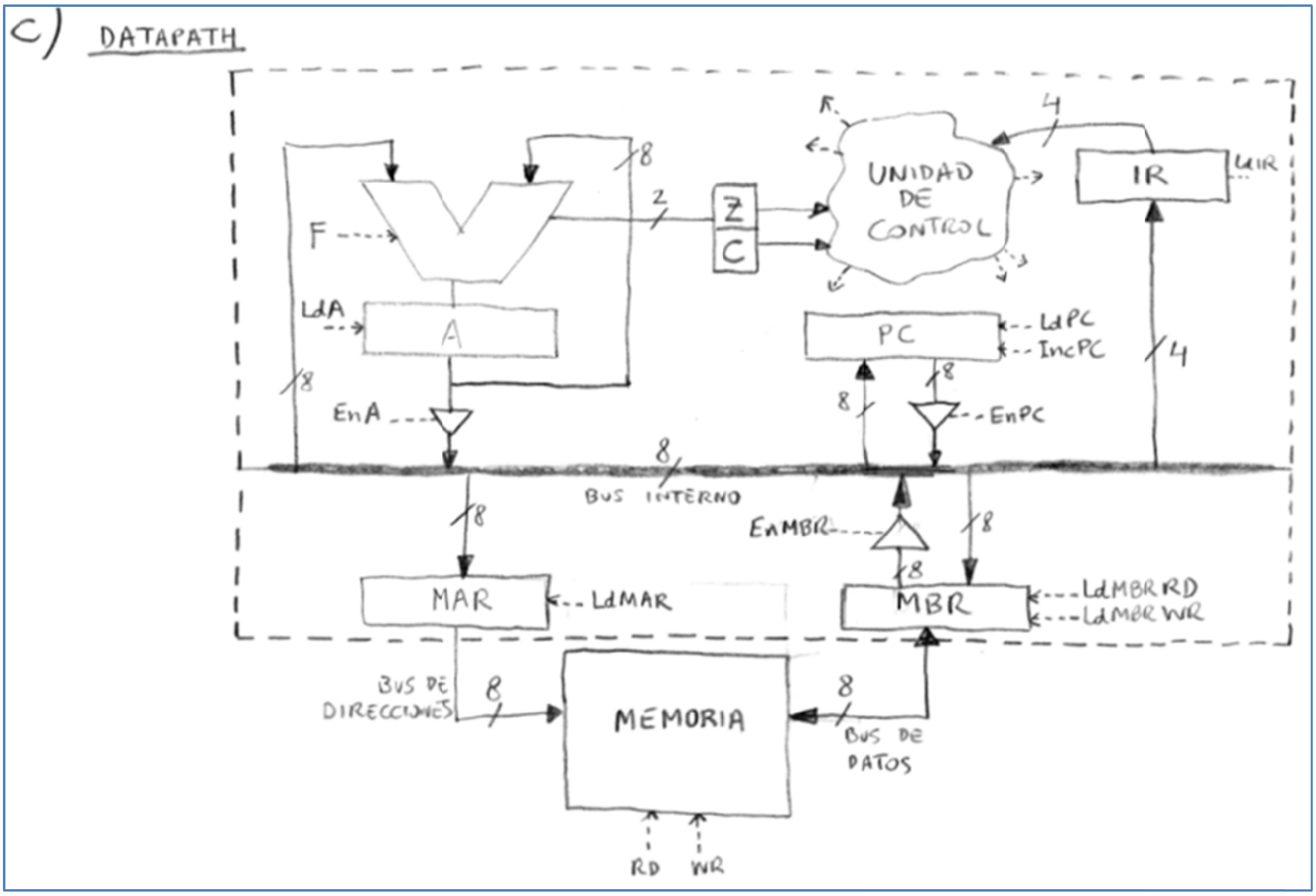
| | | | | |
|---|---|---|---|---|
| Captacion : | | | | |
| | MAR := PC ; | 0 | Sel = PC, 0 ; Op = R+S ; EnALU; LdMar ; | (1) |
| | MBR := M [ MAR ] ; | 1 | EnM ; R/Wm ; Ld MBRm ; | (2) |
| | IR := MBR ; | 2 | En MBR ; LdIR ; | (3) |
| | PC := PC+1 ; goto f(IR) | 3 | Sel = PC, 0 ; OP = R+S ; Cin ; LdPC ; | (4) |
| ADD: | RES := SUMA ; goto DosOper | 4 | | (5) |
| SUB: | RES := SUB ; goto DosOper | 5 | | (6) |
| AND: | RES := AND ; goto DosOper | 6 | | (7) |
| OR : | RES := OR ; goto DosOper | 7 | | (8) |
| NEG : | RES := NEG ; goto UnOper | 8 | | (9) |
| NOT : | RES := NOT ; goto UnOper | 9 | | (10) |
| UnOper : | SBR := S [ SP ] ; | 10 | EnS ; R/Ws ; LdSBRs ; | (11) |
| | ALU := RES ; A := ALU SBR | 11 | Op := RES ; Sel = BUS, 0 ; LdA ; | (12) |
| | SBR := A ; | 12 | Sel = 0, A ; Op = R+S ; EnALU ; LdSBRb ; | (13) |
| | S[SP] := SBR ; goto Captacion | 13 | EnS ; | (14) |
| DosOper : | SBR := S[SP] ; | 14 = 10 | | |
| | A := SBR ; SP := SP+1 ; | 15 | EnSBR ; Sel = Bus, 0 ; Op = R+S ; LdA ; Inc | (15) |
| | SBR := S [SP] ; | 16 = 10 | | |
| | ALU := RES ; A := A ALU SBR ; | 17 | Op := RES ; Sel = BUS, A ; LdA | (16) |
| | SBR := A ; | 18 = 12 | | |
| | S [SP] := SBR ; goto Captacion | 19 = 13 | | |
| PUSH : | MAR := PC ;//PC= PC+1 ; | 20 = 0 | | |
| | MBR := M [MAR] ; | 21 = 1 | | |
| | MAR := MBR ; | 22 | EnMBR ; LdMAR ; | (17) |
| | MBR := M [MAR] ; | 23 = 1 | | |
| | SBR := MBR ; SP := SP − 1 ; | 24 | EnMBR ; LdSBRb ; Dec | (18) |
| | S [SP] := SBR ; goto Captacion | 25 = 13 | VERTICAL (CODIFICATION) | (19) |
| POP : | MAR := PC ;// PC := PC+1 ; | 26 = 0 // 27 | Dos microinstrucciones; | |
| | MBR := M[MAR] ; | 28 = 1 | | |
| | MAR := MBR ;//SBR := S [SP] ; | 29 = 22 // 30 = 10 | EnMBR ; LdMAR // EnS ; R/Ws ; LdSBRs ; | |
| | MBR := SBR ; SP := SP+1 ; | 31 | EnSBR ; LdMBRb ; Inc ; | (20) |
| | M [MAR] := MBR ; goto Captacion | 32 | EnM ; | (21) |
| LDSPX : | MAR := PC ; // PC := PC+1 ; | 33 = 0 // 34 = 27 | | |
| | MBR := M [MAR] ; | 35 = 1 | | |
| | MAR := MBR ; | 36 = 22 | | |
| | MBR := M [MAR] ; | 37 = 1 | | |
| | SP := MBR ; goto Captacion | 38 | EnMBR ; LdSP ; | (22) |

## Problema UC 10.d)

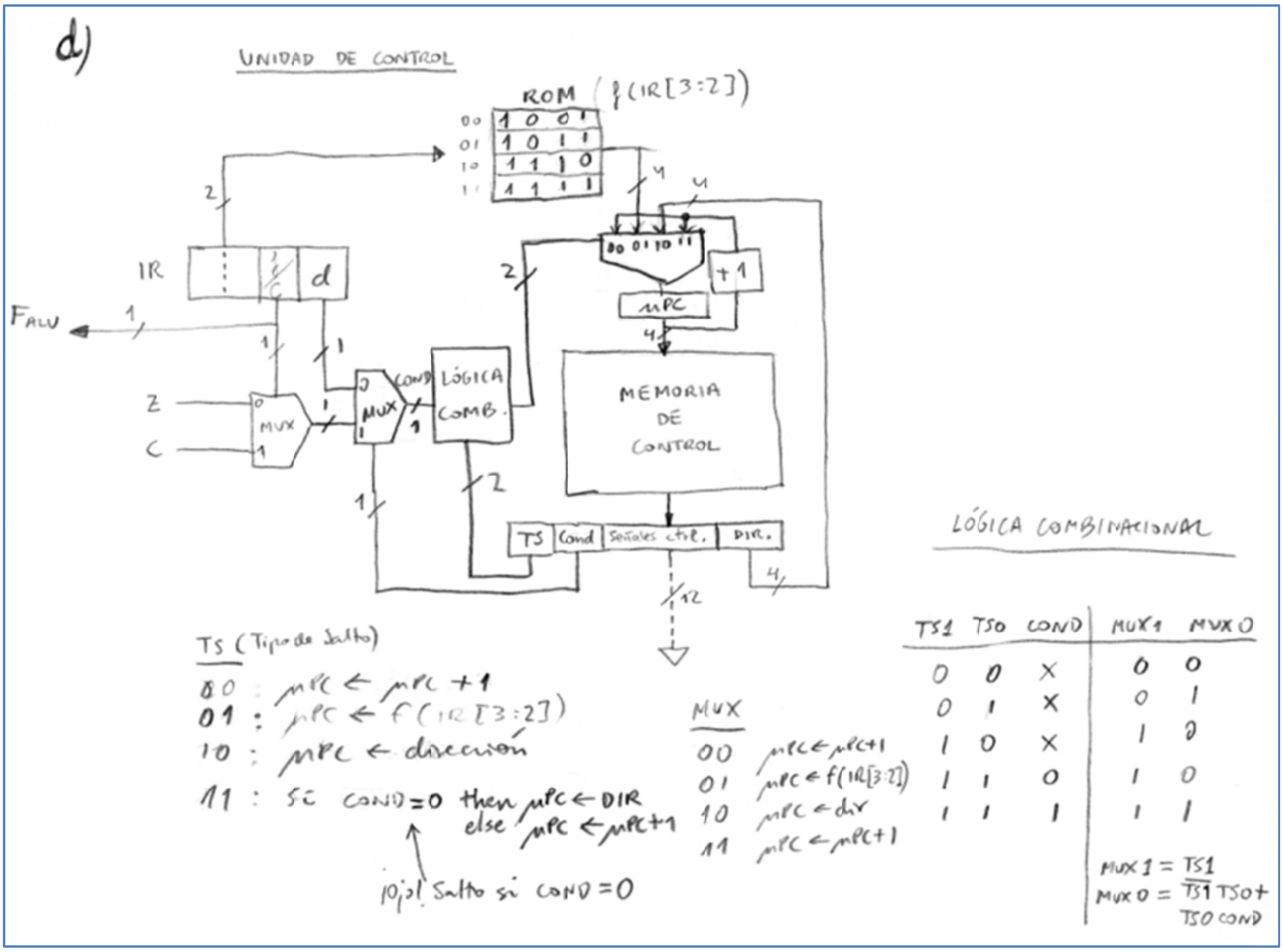


```
1    FETCH:         MAR := PC;
2                   MDR := M[MAR]; PC := PC + 1; // Captar 1er. byte
3                   IR := MDR[7:4]; AUX[3:0] := MDR[3:0]; MAR := PC;
4                   if IR[3] = 1 then goto TRES_BYTES;
5                   if IR[2] = 1 then goto DOS_BYTES;
6                   goto f(IR); // Ir a instrucciones de 1 byte
                    // Instruc. de 2 bytes (Capt. 2º byte y saltar) -------------
7    DOS_BYTES:     MDR := M[MAR]; PC := PC + 1; if COND = 1 then goto COND_TRUE;
8                   goto FETCH;
9    COND_TRUE:     AUX[11:4] := MDR;
10                  PC := PC + AUX[11:0]; goto FETCH;
                    // Captar 2º y 3er. byte de las instruc. de 3 bytes ---------
2    TRES_BYTES:    MDR := M[MAR]; PC := PC + 1; // Captar 2º byte
11                  AUX[11:4] := MDR; MAR := PC;
2                   MDR := M[MAR]; PC := PC + 1; // Captar 3er. byte
12                  AUX[19:12] := MDR;
13                  MAR := AUX;                  // Leer el operando e...
14                  MDR := M[MAR]; goto f(IR);   // ...ir a instruc. de 3 bytes
                    // Instrucciones de 1 byte ----------------------------------
15   SHR:           A := A SHR 1; goto FETCH;
16   PUSH:          SP := SP - 1; MDR := A;
17                  MAR := SP;
18   ALMACENAR_A:   M[MAR] := MDR; goto FETCH;
17   POP:           MAR := SP;
19                  MDR := M[MAR]; SP := SP + 1;
20   LOAD:          A := MDR; goto FETCH; // Ésta es de 3 bytes
17   RET:           MAR := SP;
19                  MDR := M[MAR]; SP := SP + 1;
21                  AUX[19:12] := MDR; MAR := SP;
19                  MDR := M[MAR]; SP := SP + 1;
22                  AUX[11:4] := MDR; MAR := SP;
19                  MDR := M[MAR]; SP := SP + 1;
23                  AUX[3:0] := MDR[3:0];
                    // Instrucciones de 3 bytes ---------------------------------
24   JMP:           PC := AUX; goto FETCH;
25   STORE:         MDR := A; goto ALMACENAR_A;
26   ADD:           A := A + MDR; goto FETCH;
27   SUB:           A := A - MDR; goto FETCH;
28   NAND:          A := A AND MDR;
29                  A := NOT A; goto FETCH;
30   CMP:           ALU := A - MDR; goto FETCH;
31   CALL:          SP := SP - 1; MDR := 0000 # PC[3:0];
17                  MAR := SP;
32                  M[MAR] := MDR; SP := SP - 1;
33                  MAR := SP; MDR := PC[11:4];
32                  M[MAR] := MDR; SP := SP - 1;
34                  MAR := SP; MDR := PC[19:12];
35                  M[MAR] := MDR; PC := AUX; goto FETCH;
```

## Problema UC 13.c)



## Problema UC 13.d)



| | ROM | | | |
|---|---|---|---|---|
| 00 | 1 | 0 | 0 | 1 |
| 01 | 1 | 0 | 1 | 1 |
| 10 | 1 | 1 | 1 | 0 |
| 11 | 1 | 1 | 1 | 1 |

TS (Tipo de Salto)

00 : μPC ← μPC + 1
01 : μPC ← f(IR[3:2])
10 : μPC ← dirección
11 : Si COND=0 then μPC ← DIR
else μPC ← μPC+1

¡Ojo! Salto si COND=0

MUX

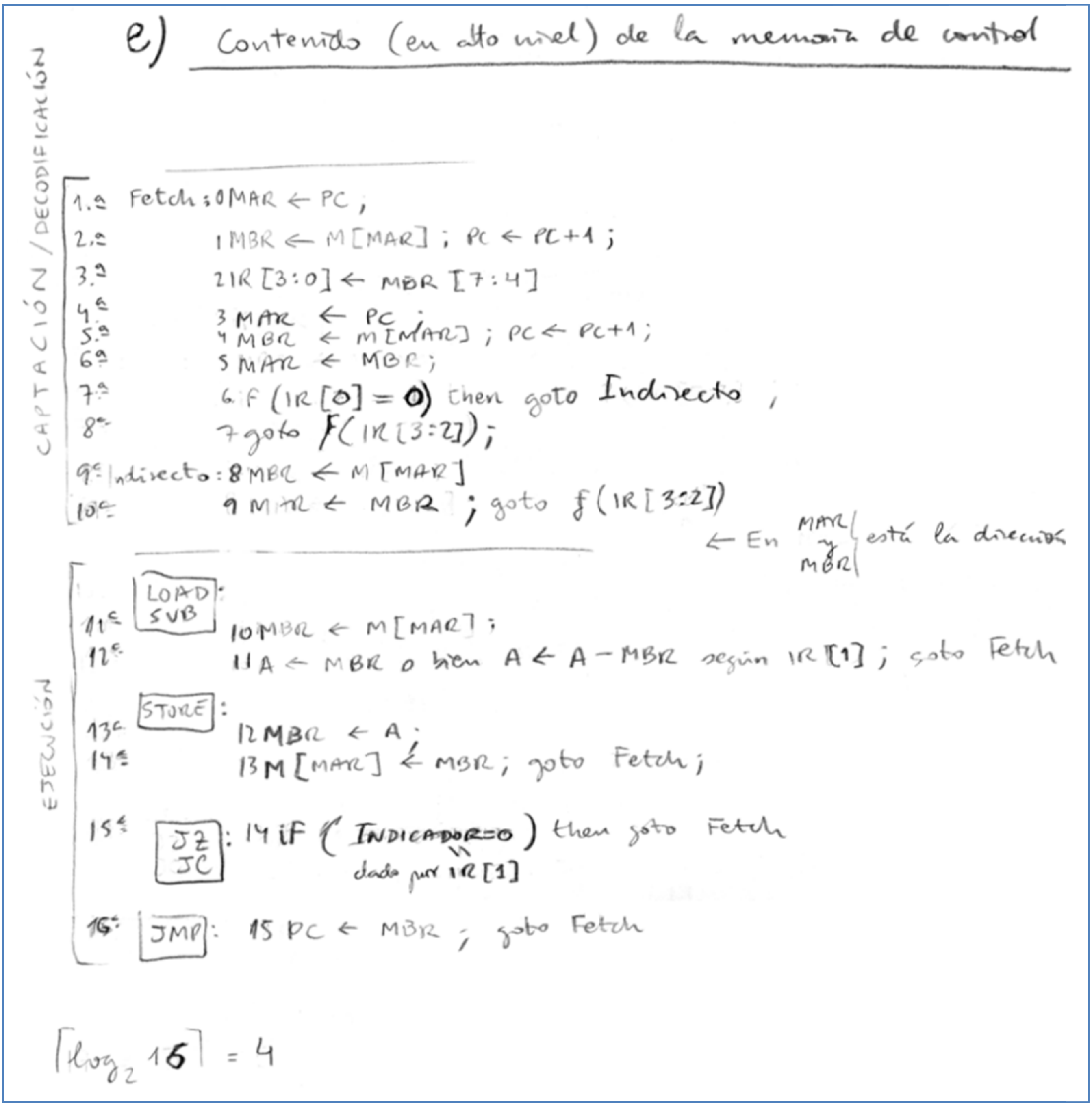
| | |
|---|---|
| 00 | μPC←μPC+1 |
| 01 | μPC←f(IR[3:2]) |
| 10 | μPC←dir |
| 11 | μPC←μPC+1 |

LÓGICA COMBINACIONAL

| TS1 | TS0 | COND | MUX1 | MUX0 |
|---|---|---|---|---|
| 0 | 0 | X | 0 | 0 |
| 0 | 1 | X | 0 | 1 |
| 1 | 0 | X | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

MUX1 = TS1
MUX0 = $\overline{TS1}$ TS0 + TS0 COND

## Problema UC 13.e)



e) Contenido (en alto nivel) de la memoria de control

CAPTACIÓN / DECODIFICACIÓN

```
1.º  Fetch: 0 MAR ← PC ;
2.º         1 MBR ← M[MAR] ; PC ← PC+1 ;
3.º         2 IR [3:0] ← MBR [7:4]
4.º         3 MAR ← PC ;
5.º         4 MBR ← M[MAR] ; PC ← PC+1 ;
6.º         5 MAR ← MBR ;
7.º         6 if (IR [0] = 0) then goto Indirecto ;
8.º         7 goto f(IR[3:2]) ;
9.º  Indirecto: 8 MBR ← M [MAR]
10.º        9 MAR ← MBR ; goto f(IR[3:2])
```

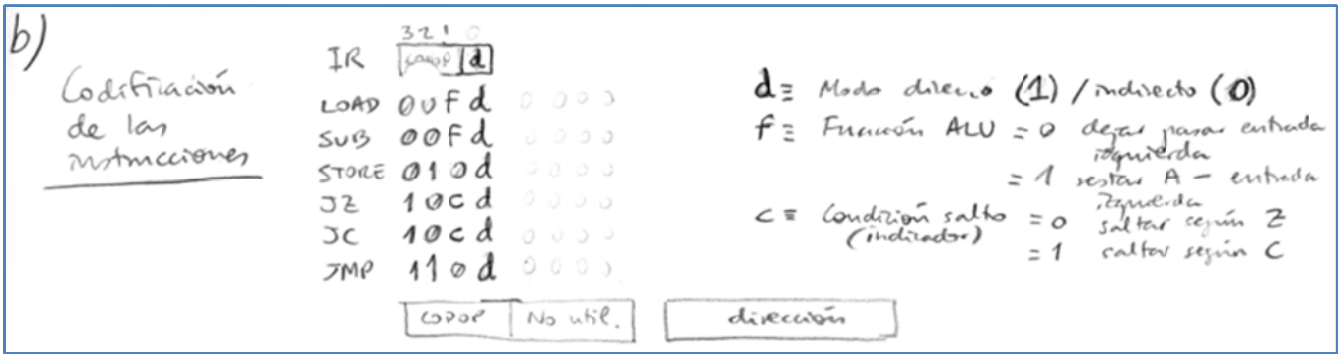
← En MAR y MBR está la dirección

EJECUCIÓN

```
     LOAD
     SUB :
11.º        10 MBR ← M[MAR] ;
12.º        11 A ← MBR o bien A ← A − MBR según IR[1] ; goto Fetch
     STORE :
13.º        12 MBR ← A ;
14.º        13 M[MAR] ← MBR ; goto Fetch ;
15.º JZ
     JC : 14 if ( INDICADOR=0 ) then goto Fetch
                 dado por IR[1]
16.º JMP :  15 PC ← MBR ; goto Fetch
```

$\lceil \log_2 16 \rceil = 4$

## Problema UC 13.b)



b) Codificación de las instrucciones

IR: 3 2 1 0 — codop | d

| | |
|---|---|
| LOAD | 0 0 f d 0 0 0 0 |
| SUB | 0 0 f d 0 0 0 0 |
| STORE | 0 1 0 d 0 0 0 0 |
| JZ | 1 0 c d 0 0 0 0 |
| JC | 1 0 c d 0 0 0 0 |
| JMP | 1 1 0 d 0 0 0 0 |

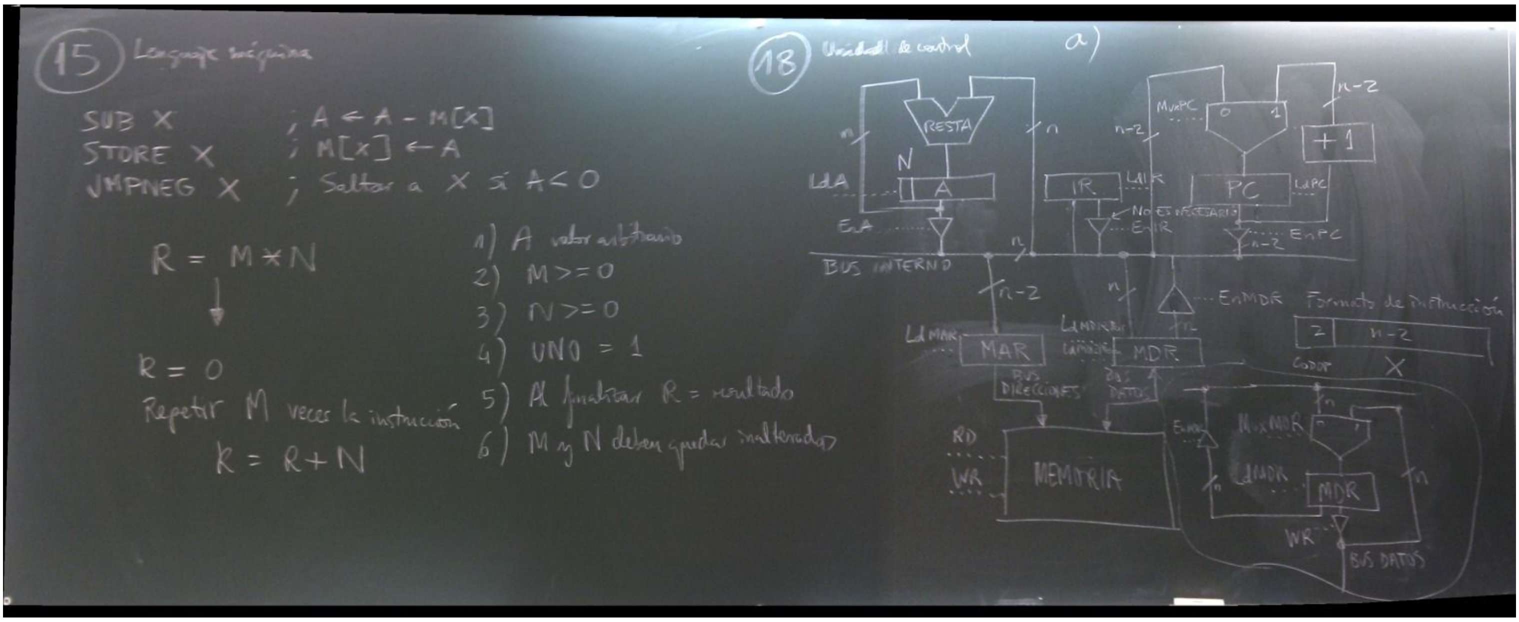
codop | No útil. | dirección

d ≡ Modo directo (1) / indirecto (0)

f ≡ Función ALU = 0 dejar pasar entrada izquierda; = 1 restar A − entrada izquierda

c ≡ Condición salto (indicador) = 0 saltar según Z; = 1 saltar según C

# Problema UC 18)



15 Lenguaje máquina

```
SUB X       ; A ← A - M[X]
STORE X     ; M[X] ← A
JMPNEG X    ; Saltar a X si A<0
```

R = M * N

↓

R = 0

Repetir M veces la instrucción

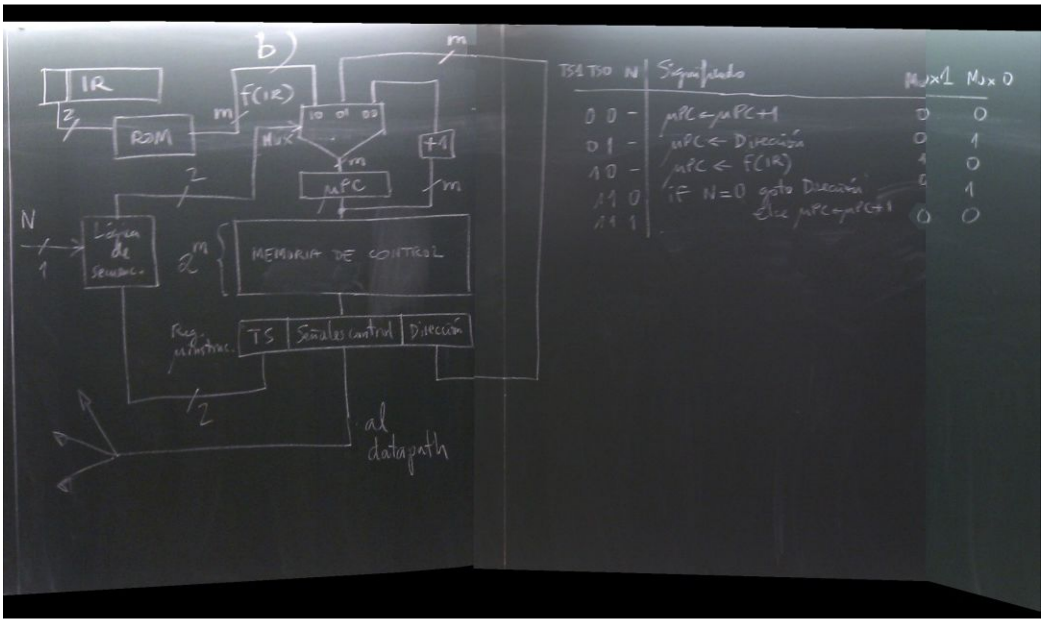
R = R+N

1) A valor arbitrario
2) M >= 0
3) N >= 0
4) UNO = 1
5) Al finalizar R = resultado
6) M y N deben quedar inalterados

18 Unidad de control   a)

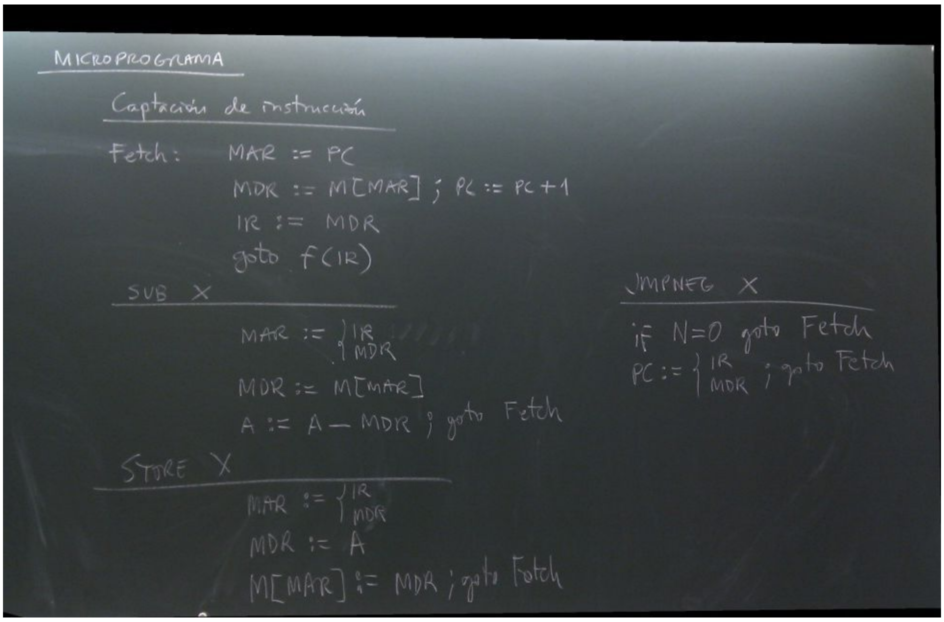
Formato de instrucción

| 2 | n-2 |
|---|---|
| CODOP | X |



b)

| TS1 | TS0 | N | Significado | Mux1 | Mux0 |
|---|---|---|---|---|---|
| 0 | 0 | - | μPC ← μPC+1 | 0 | 0 |
| 0 | 1 | - | μPC ← Dirección | 0 | 1 |
| 1 | 0 | - | μPC ← F(IR) | 1 | 0 |
| 1 | 1 | 0 | if N=0 goto Dirección | 0 | 1 |
| 1 | 1 | 1 | else μPC←μPC+1 | 0 | 0 |



MICROPROGRAMA

Captación de instrucción

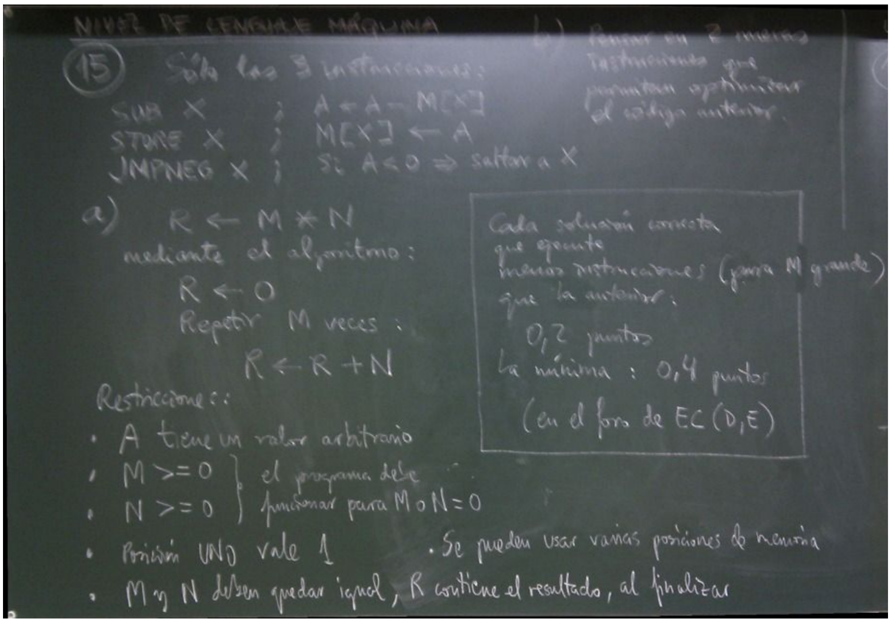
```
Fetch:  MAR := PC
        MDR := M[MAR] ; PC := PC+1
        IR := MDR
        goto f(IR)

SUB X
        MAR := { IR
                 MDR
        MDR := M[MAR]
        A := A - MDR ; goto Fetch

STORE X
        MAR := { IR
                 MDR
        MDR := A
        M[MAR] := MDR ; goto Fetch

JMPNEG X
        if N=0 goto Fetch
        PC := { IR
                MDR   ; goto Fetch
```

# Problema UC 18) versión 2



NIVEL DE LENGUAJE MÁQUINA

(15) Sólo las 3 instrucciones:

```
SUB X     ;  A ← A − M[X]
STORE X   ;  M[X] ← A
JMPNEG X  ;  Si A<0 ⇒ saltar a X
```

a) R ← M * N mediante el algoritmo:

```
R ← 0
Repetir M veces :
    R ← R + N
```

Restricciones:

- A tiene un valor arbitrario
- M >= 0 } el programa debe
- N >= 0 } funcionar para M o N = 0
- Posición UNO vale 1
- M y N deben quedar igual, R contiene el resultado, al finalizar

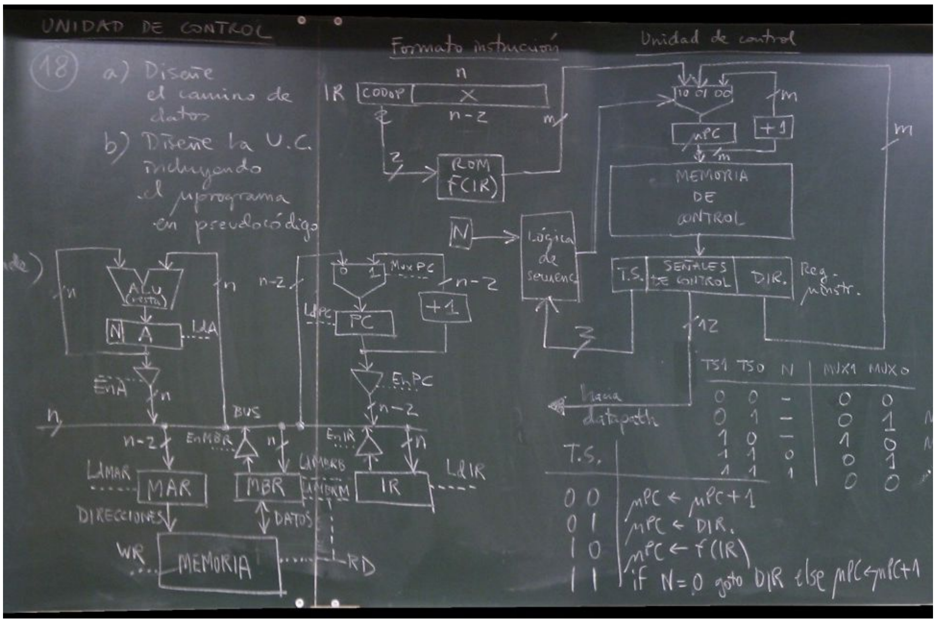
• Se pueden usar varias posiciones de memoria

b) Pensar en 2 nuevas instrucciones que permitan optimizar el código anterior.

Cada solución correcta que ejecute menos instrucciones (para M grande) que la anterior: 0,2 puntos

La mínima : 0,4 puntos

(en el foro de EC (D,E)



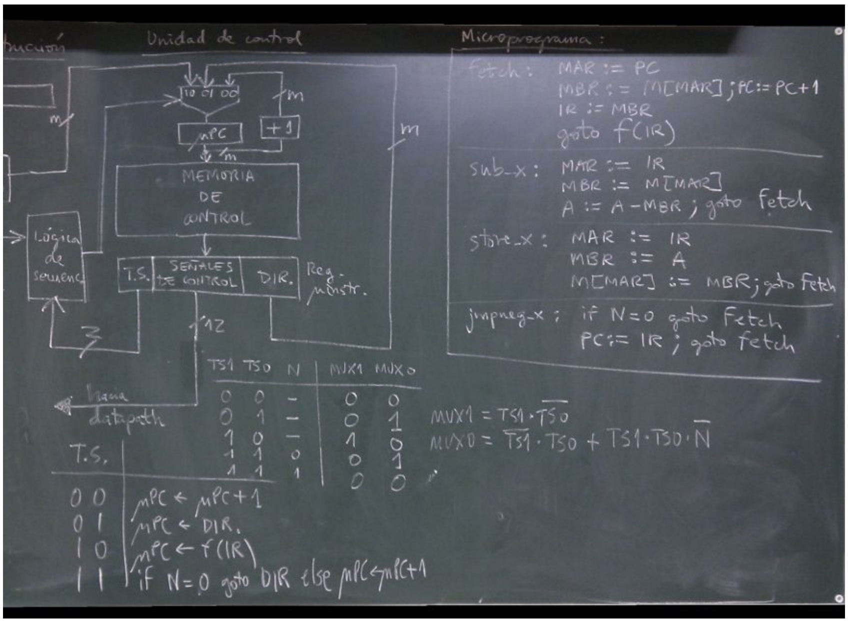
UNIDAD DE CONTROL

(18) a) Diseñe el camino de datos

b) Diseñe la U.C. incluyendo el μprograma en pseudocódigo

| T.S. | |
|---|---|
| 0 0 | μPC ← μPC+1 |
| 0 1 | μPC ← DIR. |
| 1 0 | μPC ← f(IR) |
| 1 1 | if N=0 goto DIR else μPC ← μPC+1 |

| TS1 | TS0 | N | MUX1 | MUX0 |
|---|---|---|---|---|
| 0 | 0 | – | 0 | 0 |
| 0 | 1 | – | 0 | 1 |
| 1 | 0 | – | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 |



Microprograma :

```
fetch:      MAR := PC
            MBR := M[MAR] ; PC := PC+1
            IR := MBR
            goto f(IR)

sub_x :     MAR := IR
            MBR := M[MAR]
            A := A−MBR ; goto fetch

store_x :   MAR := IR
            MBR := A
            M[MAR] := MBR ; goto Fetch

jmpneg_x ;  if N=0 goto Fetch
            PC := IR ; goto fetch
```

$MUX1 = TS1 \cdot \overline{TS0}$

$MUX0 = \overline{TS1} \cdot TS0 + TS1 \cdot TS0 \cdot \overline{N}$

| TS1 | TS0 | N | MUX1 | MUX0 |
|---|---|---|---|---|
| 0 | 0 | – | 0 | 0 |
| 0 | 1 | – | 0 | 1 |
| 1 | 0 | – | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 |

| T.S. | |
|---|---|
| 0 0 | μPC ← μPC+1 |
| 0 1 | μPC ← DIR. |
| 1 0 | μPC ← f(IR) |
| 1 1 | if N=0 goto DIR else μPC ← μPC+1 |