# Apèndix A2.

# CICLE D'INSTRUCCIÓ DEL JOC D'INSTRUCCIONS DEL PROCESSADOR EDUP12.

Aquest apèndix fa una revisió de les instruccions del processador EduP12 agrupades per tipologia d'instrucció.

S'explica el funcionament, afectació al registre d'estat i cicle d'instrucció per a cada instrucció o per tipologia d'instrucció.

La nomenclatura emprada és la mateixa de l'apèndix A1.

## A2.1. El cicle d'instrucció en EduP12

El cicle d'instrucció del joc d'instruccions d'EduP12 consta de les fases de cerca de la instrucció i execució, d'acord amb la figura A2.1

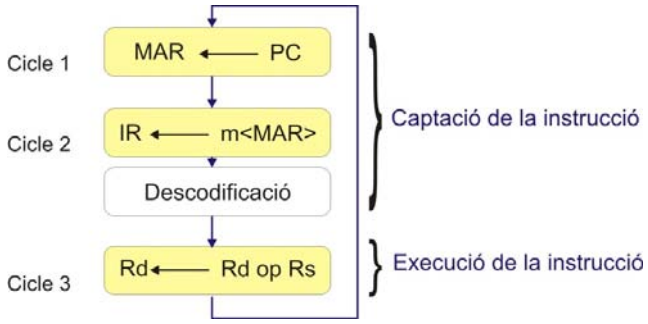


**Figura A2.1**

La fase de cerca de la instrucció necessita dos cicles de rellotge per fer la cerca, mentre que la fase d'execució pot durar més d'un cicle d'instrucció. Així com la fase de cerca és comuna a totes les instruccions, la fase d'execució és pròpia de cada grup d'instruccions o de cada instrucció.

### Fase de cerca de la instrucció

La fase de cerca de la instrucció es realitza en dos cicles de rellotge. Es realitza la següent transferència de dades:

$\Phi 1$: pMAR ← PC

$\Phi 2$: IR ← m<pMAR>, PC ← PC+1
(Descodificació)

Apèndix A2. Cicle d'instrucció del Joc d'instruccions del processador EduP12.

La figura A2.2a mostra el cablejat que s'estableix durant la fase de cerca de la instrucció.

La figura A2.2b mostra en un diagrama temporal el sincronisme entre els diferents senyals.

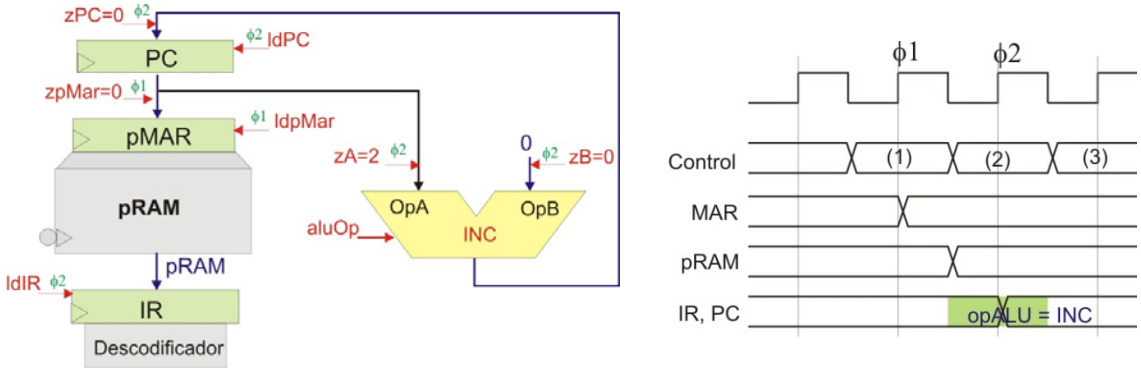


**Figura A2.2. a) Fase de cerca en la CPU. b) Diagrama temporal**

Donat que la fase de cerca de la instrucció és comuna per a totes les instruccions no s'explicitarà més.

El descodificador s'encarrega de trobar les adreces dels registres, del port d'entrada/sortida i de les constants quan cal treballar amb aquestes dades.

## A2.2. Instruccions aritmètico-lògiques de doble registre

### A2.2.1 Instrucció ADC: *Add with Carry*

- Format de la instrucció: **ADC Rd, Rs**
- Operació: Sumar el contingut dels dos registre amb el carreteig i guardar el resultat en Rd: Rd ← Rd+(Rs+C)
- Canvis en registre d'estat:

  C=1 quan l'operació produeix excés.
  Z=1 sempre que el resultat sigui 0
  N=1 sempre que el bit més significatiu sigui 1.
  V=1 quan el resultat sigui de signe diferent als operands.

### A2.2.2 Instrucció ADD: *Add*

- Format de la instrucció: **ADD Rd, Rs**
- Operació: Sumar el contingut dels dos registres i guardar el resultat en Rd: Rd ← Rd+Rs
- Canvis en registre d'estat:

  C=1 quan l'operació produeix excés.
  Z=1 sempre que el resultat sigui 0
  N=1 sempre que el bit més significatiu sigui 1.
  V=1 quan el resultat sigui de signe diferent als operands.

### A2.2.3 Instrucció AND. *Logical AND*

- Format de la instrucció: **AND Rd, Rs**
- Operació: Fer la l-lògica bit a bit de dos registres i guardar el resultat en Rd:

Rd ← Rd∧Rs

- Canvis en registre d'estat:

  C=0.
  Z=1 sempre que el resultat sigui 0
  N=1 sempre que el bit més significatiu sigui 1.
  V=0.

### A2.2.4 Instrucció CP: *Compare*

- Format de la instrucció: **CP Rd, Rs**
- Operació: Restar el contingut dels dos registres i no guardar el resultat: Rd ← Rd-Rs
- Canvis en registre d'estat:

  C=1 quan l'operació produeix excés.
  Z=1 sempre que el resultat sigui 0
  N=1 sempre que el bit més significatiu sigui 1.
  V=1 quan essent de diferent signe els operands, el resultat sigui de signe diferent al subtrahend.

### A2.2.5 Instrucció CPC: *Compare with Carry*

- Format de la instrucció: **CPC Rd, Rs**
- Operació: Restar el contingut dels dos registres amb el carreteig i no guardar el resultat: Rd ← Rd-(Rs+C)
- Canvis en registre d'estat:

  C=1 quan l'operació produeix excés.
  Z=1 sempre que el resultat sigui 0
  N=1 sempre que el bit més significatiu sigui 1.
  V=1 quan essent de diferent signe els operands, el resultat sigui de signe diferent al subtrahend.

### A2.2.6 Instrucció EOR: *Exclusive-Or*

- Format de la instrucció: **EOR Rd, Rs**
- Operació: Fer la or-exclusiva bit a bit de dos registres i guardar el resultat en Rd:

  Rd ← Rd⊕Rs

- Canvis en registre d'estat:

  C=0.
  Z=1 sempre que el resultat sigui 0
  N=1 sempre que el bit més significatiu sigui 1.
  V=0.

### A2.2.7 Instrucció MOV. *Move*

- Format de la instrucció: **MOV Rd, Rs**
- Operació: Transferir el contingut del registre Rs al Rd i guardar el resultat en Rd:

  Rd ← Rs.

- Canvis en registre d'estat: No afecta.

Apèndix A2. Cicle d'instrucció del Joc d'instruccions del processador EduP12.

### A2.2.8 Instrucció OR: *Logical OR*

- Format de la instrucció: **OR Rd, Rs**
- Operació: O-lògica bit a bit de dos registres i guardar el resultat en Rd:

  Rd ← Rd∨Rs
- Canvis en registre d'estat:

  C=0.
  Z=1 sempre que el resultat sigui 0
  N=1 sempre que el bit més significatiu sigui 1.
  V=0.

### A2.2.9 Instrucció SBC. *Substract with Carry*

- Format de la instrucció: **SBC Rd, Rs**
- Operació: Restar el contingut dels dos registres amb el carreteig i guardar el resultat en Rd: Rd ← Rd-(Rs+C)
- Canvis en registre d'estat:

  C=1 quan l'operació produeix excés.
  Z=1 sempre que el resultat sigui 0
  N=1 sempre que el bit més significatiu sigui 1.
  V=1 quan essent de diferent signe els operands, el resultat sigui de signe diferent al subtrahend.

### A2.2.10 Instrucció SUB. *Substract*

- Format de la instrucció: **SUB Rd, Rs**
- Operació: Restar el contingut dels dos registres i guarda el resultat en Rd: Rd ← Rd-Rs
- Canvis en registre d'estat:

  C=1 quan l'operació produeix excés.
  Z=1 sempre que el resultat sigui 0
  N=1 sempre que el bit més significatiu sigui 1.
  V=1 quan essent de diferent signe els operands, el resultat sigui de signe diferent al subtrahend.

### A2.2.11 Instrucció TST: *Test*

- Format de la instrucció: **TST Rd, Rs**
- Operació: Fer la I-lògica bit a bit del contingut dels dos registres i no guarda el resultat: Rd ← Rd∧Rs
- Canvis en registre d'estat:

  C=0.
  Z=1 sempre que el resultat sigui 0
  N=1 sempre que el bit més significatiu sigui 1.
  V=0.

Apèndix A2. Cicle d'instrucció del Joc d'instruccions del processador EduP12.

### A2.2.12 Fase d'execució, genèrica per les instruccions amb doble registre

La fase d'execució realitza l'operació amb les dades dels registres descodificats per la instrucció i dura un cicle de rellotge (figura A2.3). Els senyals addrRd i addrRs venen donats pel descodificador d'instruccions.

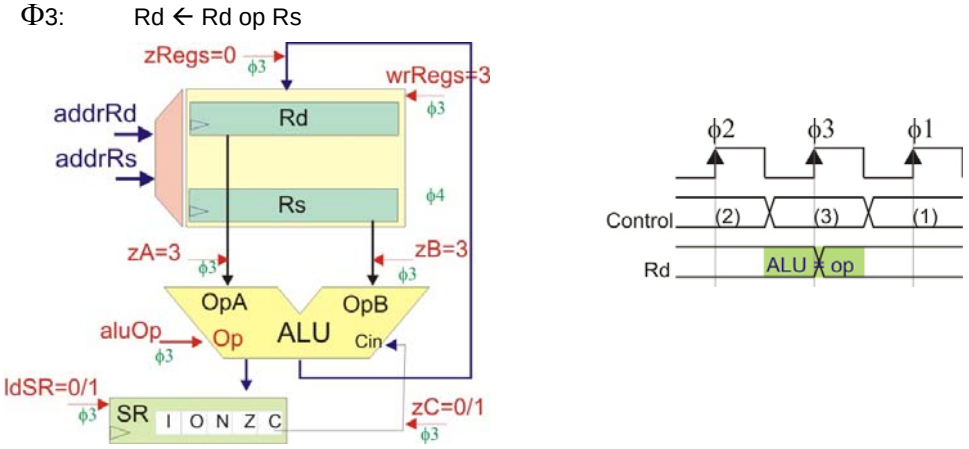


**Figura A2.3. a) Fase d'execució d'instruccions aritmètico-lògiques de doble registre. b) Diagrama temporal**

Com a excepció la instrucció MOV no actualitza el registre d'estat.

## A2.3. Instruccions aritmètico-lògiques de registre simple

### A2.3.1 Instrucció ASR: *Arithmetic Shift to Right*

- Format de la instrucció: **ASR Rd**.

  Operació: Desplaçament aritmètic a la dreta: Rd ← Rd(11) & Rd(11...1), C←Rd(0).

- Canvis en registre d'estat:

  C=Rd(0).
  Z=1 sempre que el resultat sigui 0
  N=1 sempre que el bit més significatiu sigui 1.
  V= C ⊕ N

### A2.3.2 Instrucció CLR: *Clear register*

- Format de la instrucció: **CLR Rd**
- Operació: Fer Rd ← 0.
- Implementada com a EOR Rd, Rd.

### A2.3.3 Instrucció COM: *Complement*

- Format de la instrucció: **COM Rd**
- Operació: Complement a la base disminuïda: Rd ← 0xFF-Rd
- Canvis en registre d'estat:

  C=1.
  Z=1 sempre que el resultat sigui 0

N=1 sempre que el bit més significatiu sigui 1.
V=0.

### A2.3.4 Instrucció DEC: *Decrement*

- Format de la instrucció: **DEC Rd**
- Operació: Decrementar el contingut del registre Rd: Rd ← Rd-1.
- Canvis en registre d'estat:

  C=1 quan l'operació produeix excés.
  Z=1 sempre que el resultat sigui 0
  N=1 sempre que el bit més significatiu sigui 1.
  V=1 quan el resultat sigui de signe diferent a l'operand.

### A2.3.5 Instrucció INC: *Increment*

- Format de la instrucció: **INC Rd**

  Operació: Incrementar el contingut del registre Rd: Rd ← Rd+1.
- Canvis en registre d'estat:

  C=1 quan l'operació produeix excés.
  Z=1 sempre que el resultat sigui 0
  N=1 sempre que el bit més significatiu sigui 1.
  V=1 quan el resultat sigui de signe diferent a l'operand.

### A2.3.6 Instrucció LSL: *Logical Shift to Left*

- Format de la instrucció: **LSL Rd**
- Operació: Desplaçament lògic a l'esquerra: Rd ← Rd(10...0) & 0, C ←Rd(11).
- Implementada com a ADD Rd, Rd.

### A2.3.7 Instrucció LSR: *Logical Shift to Right*

- Format de la instrucció: **LSR Rd**

  Operació: Desplaçament lògic a la dreta: Rd ← 0 & Rd(11...1), C ←Rd(0).
- Canvis en registre d'estat:

  C= Rd(0).
  Z=1 sempre que el resultat sigui 0
  N=1 sempre que el bit més significatiu sigui 1.
  V= C ⊕ N.

### A2.3.8 Instrucció NEG: *Negation*

- Format de la instrucció: **NEG Rd**
- Operació: Complement a la base: Rd ← 0x00-Rd.
- Canvis en registre d'estat:

  C=1 quan l'operació produeix excés.
  Z=1 sempre que el resultat sigui 0
  N=1 sempre que el bit més significatiu sigui 1.
  V=1 quan el resultat sigui de signe diferent als operands.

### A2.3.9 Instrucció ROL: *Rotation Left*

- Format de la instrucció: **ROL Rd**
- Operació: Rotació a l'esquerra amb carry: Rd ← Rd(10...0) & C, C ←Rd(11).
- Canvis en registre d'estat:

  C= Rd(11).
  Z=1 sempre que el resultat sigui 0
  N=1 sempre que el bit més significatiu sigui 1.
  V= C ⊕ N.

### A2.3.10 Instrucció ROR: *Rotation Right*

- Format de la instrucció: **ROR Rd**
- Operació: Rotació a la dreta amb carry: Rd ← C & Rd(11...1), C ←Rd(0).
- Canvis en registre d'estat:

  C= Rd(0).
  Z=1 sempre que el resultat sigui 0
  N=1 sempre que el bit més significatiu sigui 1.
  V= C ⊕ N.

### A2.3.11 Instrucció SWAP: *Rotació 4 bits a la dreta*

- Format de la instrucció: **SWAP Rd**
- Operació: Rotació de 4 bits a la dreta sense carry: Rd ← Rd(3...0) & Rd(11...4).
- Canvis en registre d'estat:

  C es manté.
  Z=1 sempre que el resultat sigui 0
  N=1 sempre que el bit més significatiu sigui 1.
  V= C ⊕ N.

### A2.3.12 Fase d'execució, genèrica per les instruccions d'operand simple

La fase d'execució és similar a la fase d'execució de les instruccions aritmètico-lògiques de doble registre (veure figura A2.3). Tan sols cal fer que el senyal de control $Z_B$ valgui 0. També dura un cicle de rellotge:

$\Phi 3$: Rd ← op Rd

## A2.4. Instruccions aritmètico-lògiques amb immediats

### A2.4.1 Instrucció LDI: *Load Immediat*

- Format de la instrucció: **LDI Rd, k**

  Operació: Càrrega a registre de constant: Rd ← k.
- Canvis en registre d'estat: no el modifica.

### A2.4.2 Instrucció ADDI: *Add with Immediat*

- Format de la instrucció: **ADDI Rd, k**

Apèndix A2. Cicle d'instrucció del Joc d'instruccions del processador EduP12.

- Operació: Suma de registre amb constant, amb guarda a regsitre: Rd ← Rd+k.
- Canvis en registre d'estat:

  C=1 quan l'operació produeix excés.
  Z=1 sempre que el resultat sigui 0
  N=1 sempre que el bit més significatiu sigui 1.
  V=1 quan el resultat sigui de signe diferent als operands.

### A2.4.3 Instrucció ANDI: *AND with Immediat*

- Format de la instrucció: **ANDI Rd, k**
- Operació: I-lògica bit a bit de registre amb constant, amb guarda a registre:

  Rd ← Rd∧k.
- Canvis en registre d'estat:

  C=0.
  Z=1 sempre que el resultat sigui 0
  N=1 sempre que el bit més significatiu sigui 1.
  V=0.

### A2.4.4 Instrucció CPI: *Compare with Immediat*

- Format de la instrucció: **CPI Rd, k**
- Operació: Restar de registre un valor constant, sense guarda a registre Rd: Rd ← Rd-k.
- Canvis en registre d'estat:

  C=1 quan l'operació produeix excés.
  Z=1 sempre que el resultat sigui 0
  N=1 sempre que el bit més significatiu sigui 1.
  V=1 quan essent de diferent signe els operands, el resultat sigui de signe diferent al subtrahend.

### A2.4.5 Instrucció ORI: *OR with Immediat*

- Format de la instrucció: **ORI Rd, k**
- Operació: O-lògica bit a bit de registre amb constant: Rd ← Rd∨k.
- Canvis en registre d'estat:

  C=0 quan l'operació produeix excés.
  Z=1 sempre que el resultat sigui 0
  N=1 sempre que el bit més significatiu sigui 1.
  V=0 quan el resultat sigui de signe diferent a l'operand.

### A2.4.6 Instrucció SBCI: *Substract with Immediat with Carry*

- Format de la instrucció: **SBCI Rd, k**
- Operació: Restar de registre un valor constant amb carreteig, amb guarda a registre: Rd ← Rd-(k+C).
- Canvis en registre d'estat:

  C=1 quan l'operació produeix excés.
  Z=1 sempre que el resultat sigui 0

N=1 sempre que el bit més significatiu sigui 1.
V=1 quan essent de diferent signe els operands, el resultat sigui de signe diferent al subtrahend.

### A2.4.7 Instrucció SUBI: *Substract with Immediat*

- Format de la instrucció: **SUBI Rd, k**
- Operació: Restar de registre un valor constant, amb guarda a registre Rd: Rd ← Rd-k.
- Canvis en registre d'estat:

C=1 quan l'operació produeix excés.
Z=1 sempre que el resultat sigui 0
N=1 sempre que el bit més significatiu sigui 1.
V=1 quan essent de diferent signe els operands, el resultat sigui de signe diferent al subtrahend.

### A2.2.11 Instrucció TSTI: *Test with Immediat*

- Format de la instrucció: **TSTI Rd, k**
- Operació: Fer la l-lògica bit a bit del contingut dels dos registres i no guarda el resultat: Rd ← Rd∧k
- Canvis en registre d'estat:

C=0.
Z=1 sempre que el resultat sigui 0
N=1 sempre que el bit més significatiu sigui 1.
V=0.

### A2.4.8 Fase d'execució, genèrica per les instruccions amb constants

La fase d'execució necessita de dos cicles per executar la instrucció. D'acord amb la figura A2.4 les fases són:

Φ3: pMAR ← PC, PC ← PC+1

Φ4: Rd ← Rd op m<pRAM>

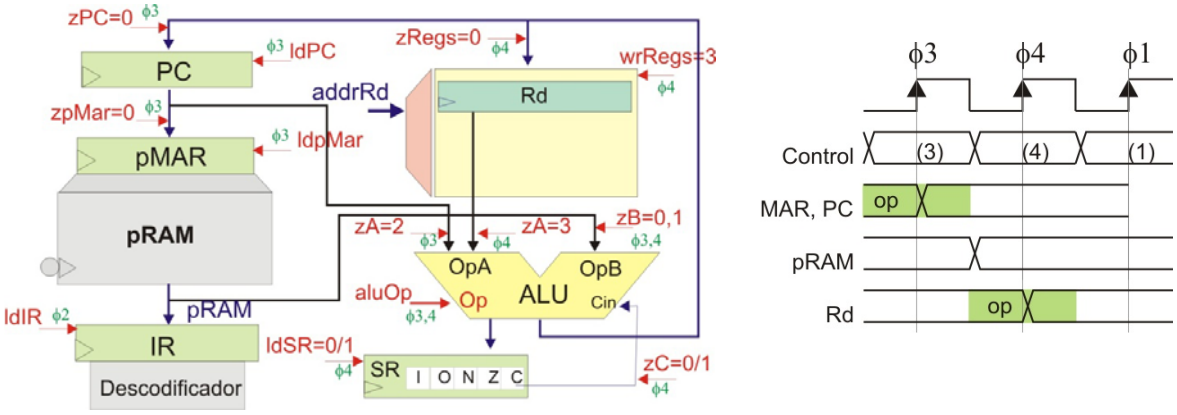


**Figura A2.4. a) Fase d'execució (requereix dos cicles de rellotge) per instruccions aritmètico-lògiques amb immediats. b) Diagrama temporal**

L'excepció al cablejat de la figura A2.4 ve donada per la instrucció LDI, que no realitza l'operació aritmètica.

## A2.5. Instruccions de salt condicional

### A2.5.1 Instrucció BRBS: *Branch if Bit is Set*

- Format de la instrucció: **BRBS b, k**
- Operació: Si ei bit de salt està activat (a 1-lògic): PC ← PC+k+1

  Si no està activat: Continuar en seqüència.
- Canvis en registre d'estat: no el modifica.

### A2.5.2 Instrucció BRBS: *Branch if Bit is Clear*

- Format de la instrucció: **BRBC b, k**
- Operació: Si ei bit de salt no està activat (a 0-lògic): PC ← PC+k+1

  Si no està activat: Continuar en seqüència.
- Canvis en registre d'estat: no el modifica.

Aquestes dues instruccions admeten instruccions específiques per preguntar sobre un bit concret. La taula A2.1 resumeix les instruccions acceptades per la màquina:

| Instrucció | Operació | Condició sobre bit | Bit (b) |
|---|---|---|---|
| **BRCC** k | Branch if Carry is Clear | C = 0 | 0 |
| **BRCS** k | Branch if Carry is Set | C = 1 | 0 |
| **BRSH** k | Branch if Same or Higher | C = 0 | 0 |
| **BRLO** k | Branch if Lower | C = 1 | 0 |
| **BRNZ (BRNE)** k | Branch if Non Zero | Z = 0 | 1 |
| **BRZE (BREQ)** k | Branch if Zero | Z = 1 | 1 |
| **BRPL** k | Branch if Plus | N = 0 | 2 |
| **BRMI** k | Branch if Minus | N = 1 | 2 |
| **BRVC** k | Branch if Overflow is Clear | V = 0 | 3 |
| **BRVS** k | Branch if Carry Set | V = 1 | 3 |
| **BRID** k | Branch if Interrupt Disabled | I = 0 | 7 |
| **BRIE** k | Branch if Interrupt Enabled | I = 1 | 7 |

**Taula A2.1 Instruccions de salt condicional sobre bit particular**

### A2.5.3 Fase d'execució

La fase d'execució és realitza en un únic cicle.

$\Phi 3$: SaltCondicional=1 ? PC ← PC+k : continuar

Així, la fase d'execució de la instrucció de salt condicional és continuar en seqüència quan no hi ha salt. Quan hi salt es tracta com si fos un salt incondicional relatiu des de la posició actual, i la seva execució és igual a la de la instrucció RJMP (figura A2.8).

## A2.6. Instruccions de salt incondicional

### A2.6.1 Instrucció ICALL: *Indirect call*

- Format de la instrucció: **ICALL**

Apèndix A2. Cicle d'instrucció del Joc d'instruccions del processador EduP12.

- Operació: Salta a la subrutina que es troba en la posició de memòria indicada pel registre Z, al mateix temps que guarda l'adreça (que es troba en PC) de retorn:

  Pila ← PC, PC ← Z, SP ← SP+1.

- El nombre Z pot ser positiu o negatiu, expressat en complement a la base.
- Canvis en registre d'estat: no el modifica.

**Fase d'execució**

- La figura A2.5 mostra la fase d'execució. Les taques que es realitzen durant els dos cicles de la fase d'execució són:

  Φ3: dMAR ← SP, SP ← SP-1

  dRAM ← PC (s'executa durant el flanc de baixada del rellotge)

  Φ4: PC ← (Z)

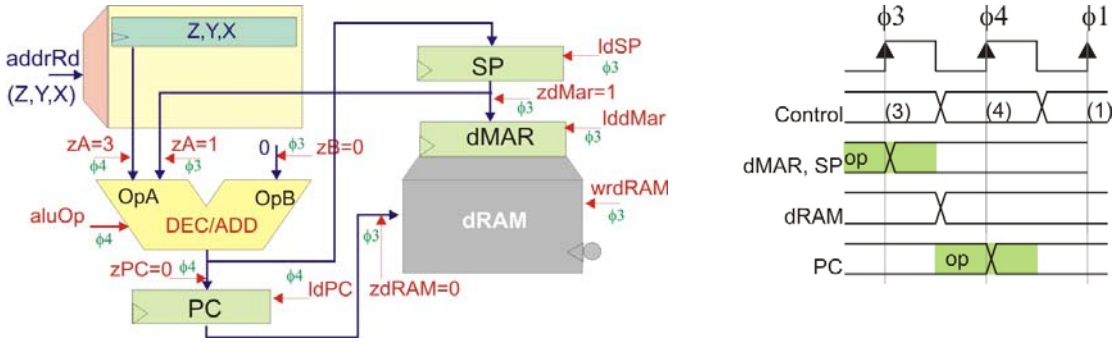


**Figura A2.5. a) Fase d'execució de la instrucció ICALL. b) Diagrama temporal**

## A2.6.2 Instrucció IJMP: *Indirect Jump*

- Format de la instrucció: **IJMP**
- Operació: Salta a la posició de memòria indicada pel registre Z: PC ← Z
- El nombre Z pot ser positiu o negatiu, expressat en complement a la base.
- Canvis en registre d'estat: no el modifica.

**Fase d'execució**

- La figura A2.6 mostra la fase d'execució. S'executa l'operació:

  Φ3: PC ← (Z)

  Que es correspon amb el segon cicle de la fase d'execució de la instrucció ICALL.

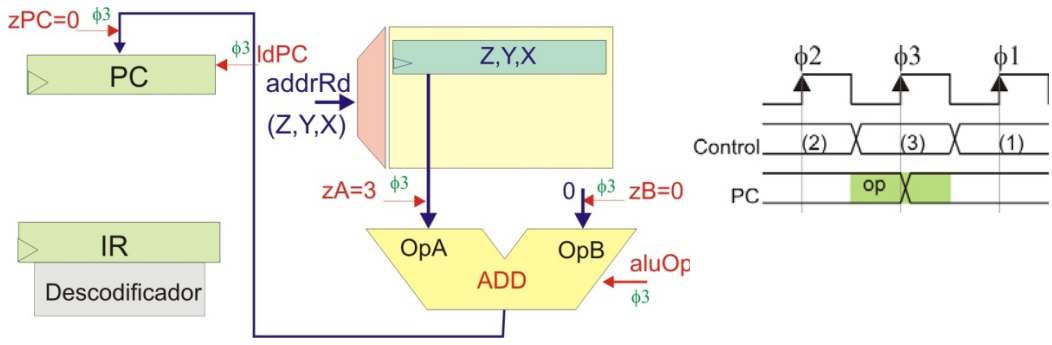


**Figura A2.6. a) Fase d'execució del la instrucció IJMP. b) Diagrama temporal**

### A2.6.3 Instrucció RCALL: *Relative Call*

- Format de la instrucció: **RCALL, k**
- Operació: Salta a la subrutina que es troba a k posicions de memòria, al mateix temps que guarda l'adreça (que es troba en PC) de retorn:

  Pila ← PC, PC ← PC+k+1, SP ← SP+1.
- El nombre k pot ser positiu o negatiu, expressat en complement a la base.
- Canvis en registre d'estat: no el modifica.

**Fase d'execució**

- La figura A2.7 mostra la fase d'execució. La fase d'execució dura dos cicles de rellotge:

  Φ3: dMAR ← SP, SP ← SP-1

  dRAM ← PC (s'executa durant el flanc de baixada del rellotge)

  Φ4: PC ← PC+k

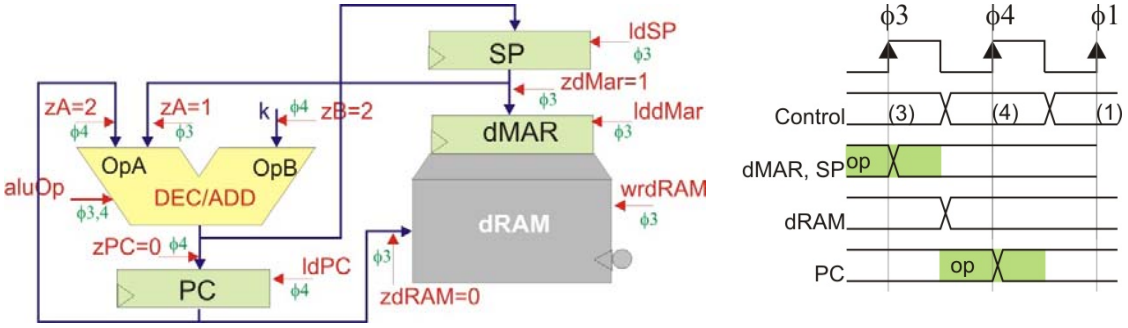


**Figura A2.7. a) Fase d'execució de la instrucció RACLL. b) Diagrama temporal**

### A2.6.4 Instrucció RJMP: *Relative Jump*

- Format de la instrucció: **RJMP k**
- Operació: Salta a k posicions de memòria: PC ← PC+k+1
- El nombre k pot ser positiu o negatiu, expressat en complement a la base.
- Canvis en registre d'estat: no el modifica.

**Fase d'execució**

- La figura A2.8 mostra la fase d'execució.

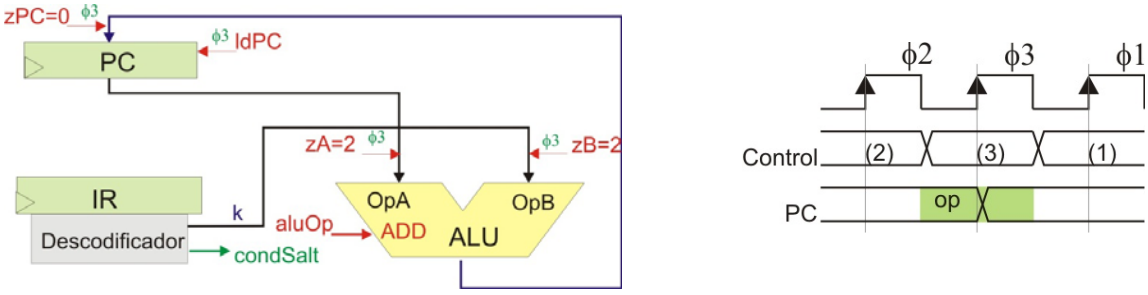


**Figura A2.8. a) Fase d'execució per la instrucció RJMP. b) Diagrama temporal**

### A2.6.5 Instrucció RET: *Return*

- Format de la instrucció: **RET**
- Operació: Retorna de subrutina. Cerca l'adreça de retorn de la pila:

  SP ← SP-1, PC ← Pila.
- Canvis en registre d'estat: no el modifica.

**Fase d'execució**

- La figura A2.6 mostra la fase d'execució.
- La fase d'execució dura dos cicles de rellotge:

  Φ3: dMAR ← (SP+1)

  Φ4: PC ← <dRAM(dMAR)>

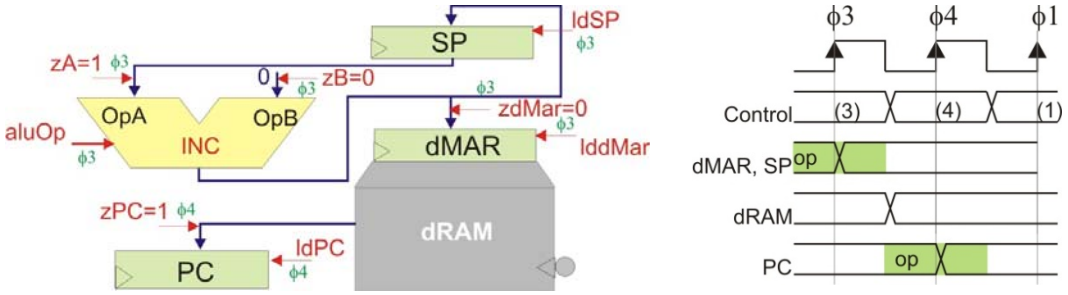


**Figura A2.9. a) Fase d'execució de la instrucció RET. b) Diagrama temporal**

### A2.6.6 Instrucció RETI: *Return from Interrupt*

- Format de la instrucció: **RETI**
- Operació: Retorna de subrutina d'interrupció. Cerca l'adreça de retorn de la pila:

  SP ← SP-1, PC ← Pila, Habilita possibilitat d'interrupció.
- Canvis en registre d'estat: no el modifica.

**Fase d'execució**

La fase d'execució és comuna a la de la instrucció RET. Tan sols s'ha de tenir en compta de reactivar la capacitat d'interrupció donat que se surt del tractament d'una rutina de servei d¡interrupció.

Φ3: dMAR ← (SP+1), ackISR ← 0

Φ4: PC ← <dRAM(dMAR)>

## A2.7. Instruccions de càrrega amb memòria de programa

### A2.7.1 Instrucció LPM: *Load from Program Memory*

- Format de la instrucció: **LPM Rd, Z** (o Y, o X)
- Operació: La instrucció carrega en un registre una dada de memòria de programa adreçada pel registre Z: Rd ← memòriaPrograma(Z) (o Y o X)
- Canvis en registre d'estat: no el modifica.

**Fase d'execució**

- La figura A2.6 mostra la fase d'execució de la instrucció LPM. S'executa en dos cicles de rellotge:

  Φ3: pMAR ← Z (o X o Y)

  Φ4: Rd ← pRAM<pMAR>

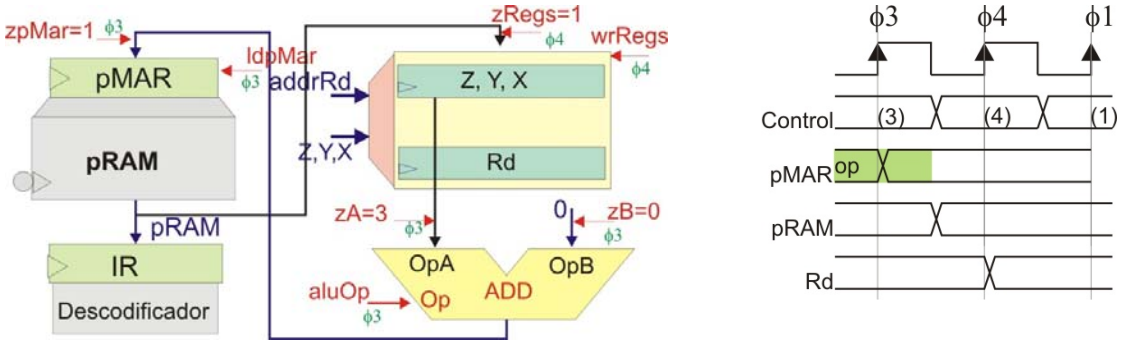


**Figura A2.10. a) Fase d'execució d'instruccins LPM. b) Diagrama temporal**

### A2.7.2 Instrucció LPM +: *Load from Program Memory with pre-Increment*

- Format de la instrucció: **LPM Rd, +Z** (o Y, o X)
- Operació: La instrucció carrega en un registre una dada de memòria de programa adreçada pel registre Z prèviament incrementat. Es realitzen l'operació:

  Rd ← memòriaPrograma(Z+1) (o Y o X)

- Canvis en registre d'estat: no el modifica.

**Fase d'execució**

- La fase d'execució és la mateixa que per la instrucció LPM (figura A2.10), tan sols realitzant el canvi de l'operació de la UAL en la fase Φ3, incrementant Z (enlloc de la suma).
- La fase d'execució dura, per tant, els dos cicles següents:

  Φ3: pMAR ← (Z+1) (o amb els registres X o Y)

  Φ4: Rd ← pRAM<pMAR>

### A2.7.3 Instrucció LPM -: *Load from Program Memory with pre-Decrement*

- Format de la instrucció: **LPM Rd, -Z** (o Y, o X)
- Operació: La instrucció carrega en un registre una dada de memòria de programa adreçada pel registre Z prèviament decrementat. Es realitzen dues operacions:

  Rd ← memòriaPrograma(Z-1) (o Y o X)

- Canvis en registre d'estat: no el modifica.

**Fase d'execució**

- La fase d'execució és la mateixa que per la instrucció LPM (figura A2.10), tan sols realitzant el canvi de l'operació de la UAL en la fase Φ3, decrementant Z (enlloc de la suma).
- La fase d'execució dura, per tant, els dos cicles següents:

  Φ3: pMAR ← (Z-1) (o amb els registres X o Y)

Apèndix A2. Cicle d'instrucció del Joc d'instruccions del processador EduP12.

Φ4: Rd ← pRAM<pMAR>

## A2.8. Instruccions de càrrega amb memòria de dades

### A2.8.1 Instrucció LDS: *Load Direct from Data Memory*

- Format de la instrucció: **LD Rd, K**
- Operació: La instrucció carrega en un registre la dada de memòria de dades adreçada per K: Rd ← memòriaDades(K)
- Canvis en registre d'estat: no el modifica.

#### Fase d'execució

La figura A2.11 mostra la fase d'execució de LDS. Necessita tres cicles de rellotge, que realitzen les següents operacions:

Φ3: pMAR ← PC, PC ← PC+1

Φ4: dMAR ← k (=<pRAM passant per ALU)

Φ5: Rd ← dRAM<dMAR>

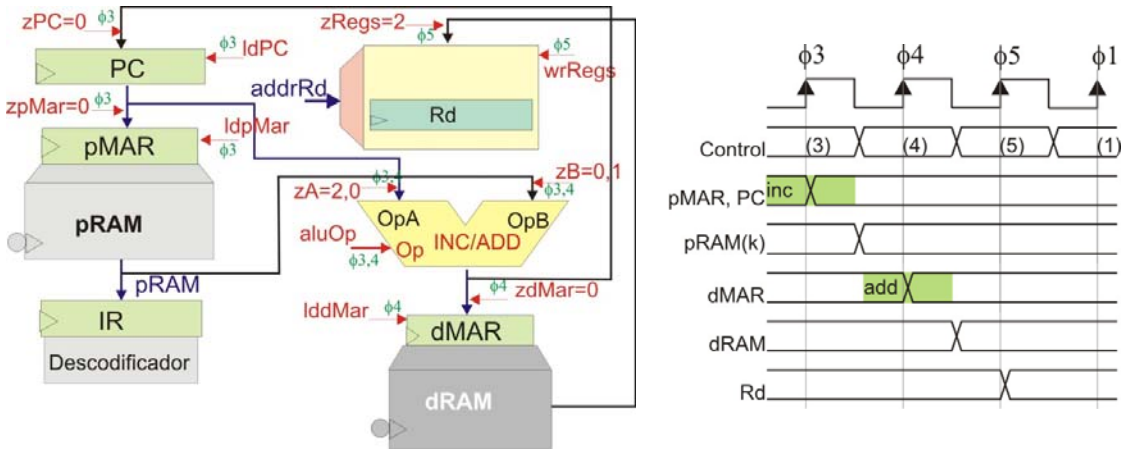


**Figura A2.11. a) Fase d'execució per instruccions de càrrega directa. b) Diagrama temporal**

### A2.8.2 Instrucció LD: *Load Indirect from Data Memory*

- Format de la instrucció: **LD Rd, Z** (o amb els registres Y o X)
- Operació: La instrucció carrega en un registre una dada de memòria de dades adreçada pel registre Z (o Y, o X): Rd ← memòriaDades(Z), (o amb els registres Y o X)
- Canvis en registre d'estat: no el modifica.

#### Fase d'execució

En la figura A2.12 es mostra la fase d'execució de les instruccions de càrrega indirecta. Es necessiten dos cicles de rellotge i es realitzen les següents operacions:

Φ3: dMAR ← (Z), o dels registres Y o Z.

Φ4: Rd ← dRAM<dMAR>

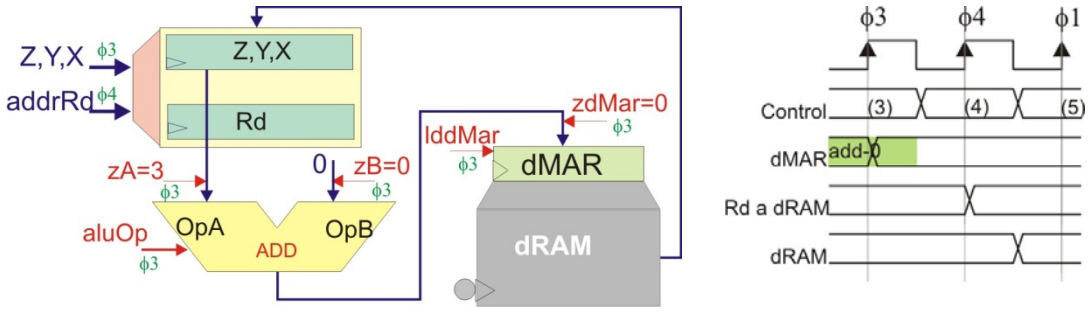


**Figura A2.12. a) Fase d'execució per instruccions de càrrega indirecta. b) Diagrama temporal**

### A2.8.3 Instrucció LD +: *Load Indirect from Data Memory with pre-Increment*

- Format de la instrucció: **LD Rd, +Z** (o amb els registres Y o X)
- Operació: La instrucció carrega en un registre una dada de memòria de dades adreçada pel registre Z (o Y, o X) prèviament incrementat. L'operació és

  Rd ← memòriaDades(Z+1) (o amb els registres Y o X)
- Canvis en registre d'estat: no el modifica.

### Fase d'execució

La fase d'execució és la mateixa que per la instrucció LD (figura A2.12). Donat que l'única diferència és el pre-increment, s'aprofita el pas per la ALU de la dada provinent del registre (Z, Y o X) per incrementar-la.

### A2.8.4 Instrucció LD -: *Load Indirect from Data Memory with pre-Decrement*

- Format de la instrucció: **LD Rd, -Z** (o amb els registres Y o X)
- Operació: La instrucció carrega en un registre una dada de memòria de dades adreçada pel registre Z (o Y, o X) prèviament decrementat. L'operació és

  Rd ← memòriaDades(Z-1) (o amb els registres Y o X)
- Canvis en registre d'estat: no el modifica.

### Fase d'execució

La fase d'execució és la mateixa que per la instrucció LD (figura A2.12). Donat que l'única diferència és el pre-decrement, s'aprofita el pas per la ALU de la dada provinent del registre (Z, Y o X) per decrementar-la.

### A2.8.5 Instrucció LDD: *Load Indirect from Data Memory with Displacement*

- Format de la instrucció: **LDD Rd, Z+q** (o amb el registre Y)
- Operació: La instrucció carrega en un registre la dada de memòria de dades adreçada pel registre Z desplaçat en q posicions, on q és un natural. Es realitza l'operació:

  Rd ← memòriaDades(Z+q) (o amb Y)
- Canvis en registre d'estat: no el modifica.

### Fase d'execució

La fase d'execució és la mateixa que per la instrucció LD (figura A2.12). Donat que l'única diferència és el desplaçament, s'aprofita el pas per la ALU de la dada provinent del registre (Z, Y o X) per sumar el desplaçament q.

## A2.8.6 Instrucció STS: *Store Direct to Data Memory*

- Format de la instrucció: **STS K, Rs**
- Operació: Guarda el contingut del registre Rs en la posició de memòria de dades adreçada per K: memòriaDades(K) ← Rs
- Canvis en registre d'estat: no el modifica.

### Fase d'execució

La figura A2.13 mostra la fase d'execució de LDS. Necessita tres cicles de rellotge, que realitzen les següents operacions:

Φ3: pMAR ← PC, PC ← PC+1

Φ4: dMAR ← k (=<pRAM passant per ALU)

Φ5: dRAM<dMAR> ← Rd

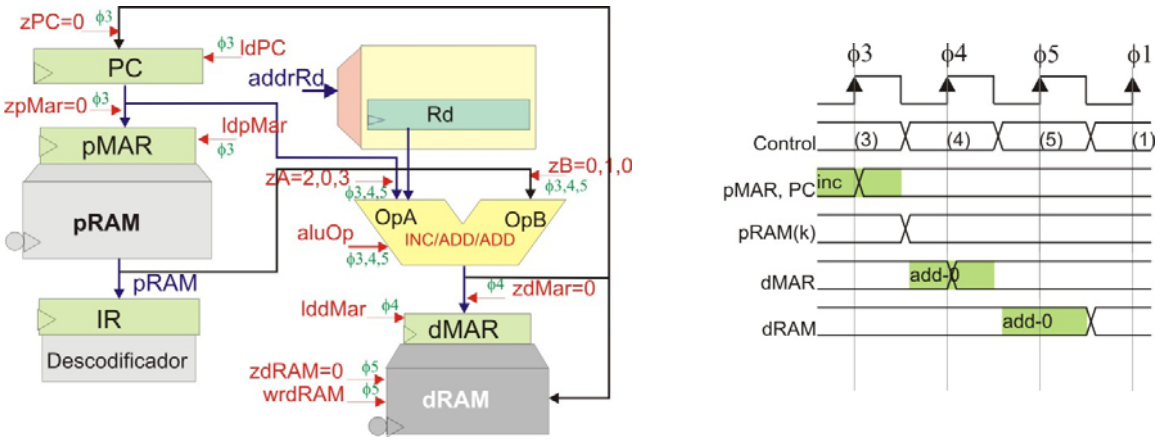


**Figura A2.13. a) Fase d'execució de la instrucció de guarda directa. b) Diagrama temporal**

## A2.8.7 Instrucció ST: *Store Indirect to Data Memory*

- Format de la instrucció: **ST Z, Rs** (o amb els registres Y o X)
- Operació: Guarda el contingut del registre Rs en la posició de memòria de dades adreçada pel registre Z (o Y, o X): memòriaDades(Z) ← Rs, (o amb els registres Y o X)
- Canvis en registre d'estat: no el modifica.

### Fase d'execució

La figura A2.14 mostra la fase d'execució de LDS. Necessita tres cicles de rellotge, que realitzen les següents operacions:

Φ3: pMAR ← PC, PC ← PC+1

Φ4: dMAR ← k (=<pRAM passant per ALU)

Φ5: Rd ← dRAM<dMAR>

Apèndix A2. Cicle d'instrucció del Joc d'instruccions del processador EduP12.

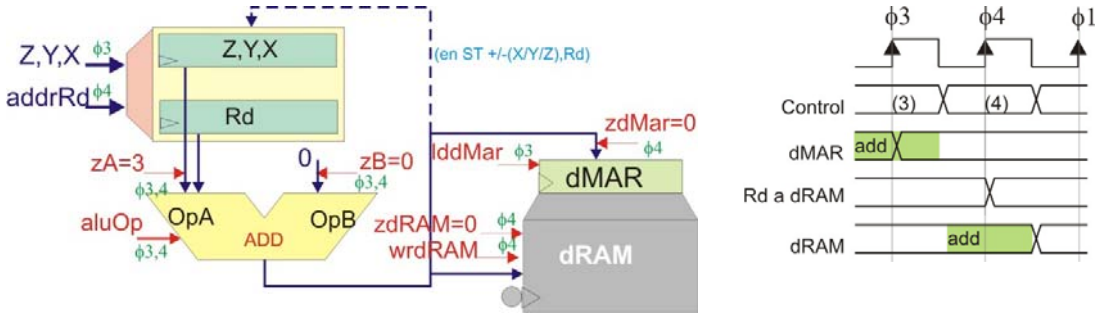


**Figura A2.14. a) Fase d'execució per instruccions de guarda indirecta. b) Diagrama temporal**

### A2.8.8 Instrucció ST +: *Store Indirect to Data Memory with pre-Increment*

- Format de la instrucció: **ST Rs, +Z** (o amb els registres Y o X)
- Operació: Guarda el contingut del registre Rs en la posició de memòria de dades adreçada pel registre Z (o Y, o X) prèviament incrementat. L'operació és

  memòriaDades(Z+1) ← Rs (o amb els registres Y o X)
- Canvis en registre d'estat: no el modifica.

### Fase d'execució

La fase d'execució és la mateixa que per la instrucció LD (figura A2.12). Donat que l'única diferència és el pre-increment, s'aprofita el pas per la ALU de la dada provinent del registre (Z, Y o X) per incrementar-la.

### A2.8.9 Instrucció ST -: *Store Indirect to Data Memory with pre-Decrement*

- Format de la instrucció: **ST –Z, Rs** (o amb els registres Y o X)
- Operació: Guarda el contingut del registre Rs en la posició de memòria de dades adreçada pel registre Z (o Y, o X) prèviament decrementat. L'operació és

  memòriaDades(Z-1) ← Rs (o amb els registres Y o X)
- Canvis en registre d'estat: no el modifica.

### Fase d'execució

La fase d'execució és la mateixa que per la instrucció LD (figura A2.12). Donat que l'única diferència és el pre-decrement, s'aprofita el pas per la ALU de la dada provinent del registre (Z, Y o X) per decrementar-la.

### A2.8.10 Instrucció STD: *Store Direct to Data Memory with Displacement*

- Format de la instrucció: **STD Z+q, Rs** (o amb el registre Y)
- Operació: Guarda el contingut del registre Rs en la posició de memòria de dades adreçada pel registre Z desplaçat en q posicions, on q és un natural. Es realitza l'operació: memòriaDades(Z+q) ← Rs (o amb Y)
- Canvis en registre d'estat: no el modifica.

### Fase d'execució

La fase d'execució és la mateixa que per la instrucció LD (figura A2.12). Donat que l'única diferència és la suma del desplaçament, s'aprofita el pas per la ALU de la dada provinent del registre (Z, Y o X) per realitzar-la.

## A2.9. Instruccions d'entrada/sortida

### A2.9.1 Instrucció IN: *Input from PORT*

- Format de la instrucció: **IN Rd, PORT**
- Operació: Entra a Rd una dada d'un port d'entrada de la memòria d'entrada/sortida: Rd ← PORT. La instrucció conté, com a camps, l'adreça del registre d'entrada i l'adreça de la posició de memòria corresponent al perifèric d'entrada. Ambdues adreces es descodifiquen en la fase Φ2.

  La memòria d'entrada/sortida disposa de 64 posicions
- Canvis en registre d'estat: no es modifica.

### Fase d'execució

La figura A2.15 mostra la fase d'execució de la instrucció entrada de port. La UCP s'encarrega, senzillament, de llegir del port d'entrada.

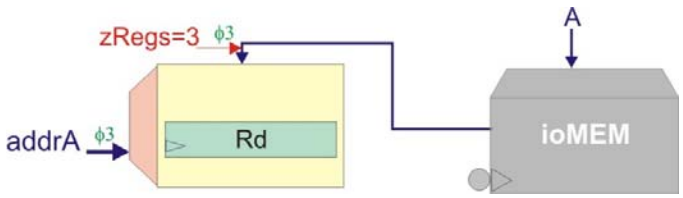


**Figura A2.15. a) Fase d'execució per instruccions d'entrada de port.**

Els registres de memòria d'entrada actualitzen les dades en flanc de baixada.

Es realitza l'operació:

Φ3: Rd ← ioMem<ioAddr>

### A2.9.2 Instrucció OUT: *Output to PORT*

- Format de la instrucció: **OUT PORT, Rs**
- Operació: Posa en un port de sortida de la memòria d'entrada/sortida el contingut de Rs: PORT ← Rs

  La memòria d'entrada/sortida disposa de 64 posicions
- Canvis en registre d'estat: no el modifica.

### Fase d'execució

La figura A2.16 mostra la fase d'execució de la instrucció de sortida de port. La UCP s'encarrega, senzillament, d'enviar la dada cap al port de sortida.

Apèndix A2. Cicle d'instrucció del Joc d'instruccions del processador EduP12.

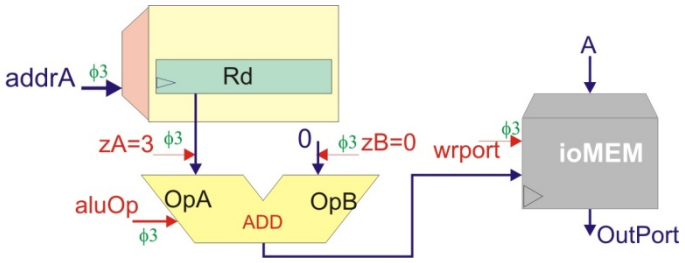


**Figura A2.16. a) Fase d'execució per instruccions de sortida a port.**

Es realitza l'operació:

Φ3: ioMem<ioAddr> ← Rs

## A2.10. Altres instruccions

### A2.10.1 Instrucció PUSH: *Push to Stack*

- Format de la instrucció: **PUSH Rs**
- Operació: Guarda en pila, en la posició apuntada per l'apuntador de pila, el contingut de Rd. Després decrementa l'apuntador a pila. Són dues operacions:

  Pila ← Rs
  SP ← SP-1
- Canvis en registre d'estat: no el modifica.

### Fase d'execució

La fase necessita dos cicles de rellotge (figura A2.17).

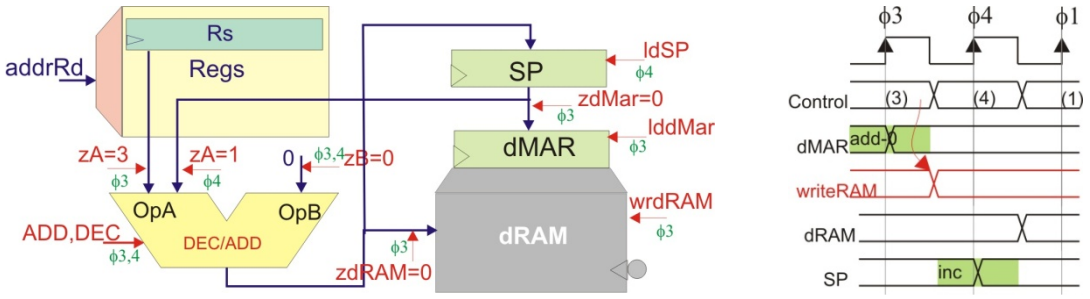


**Figura A2.17. a) Fase d'execució per instruccions de guarda a pila. b) Diagrama temporal**

### A2.10.2 Instrucció POP: *Pop from Stack*

- Format de la instrucció: **POP Rd**
- Operació: Primer incrementa l'apuntador a pila i després recupera a Rd la dada que es troba en la pila. Són dues operacions:

  SP ← SP+1
  Rd ← Pila
- Canvis en registre d'estat: no el modifica.

Apèndix A2. Cicle d'instrucció del Joc d'instruccions del processador EduP12.

### Fase d'execució

La figura A2.18 mostra la fase d'execució que es realitza en dos cicles de rellotge.

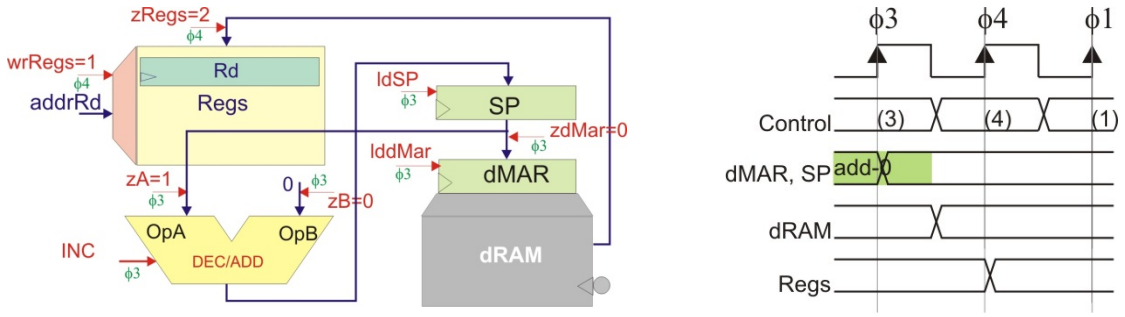


**Figura A2.18. a) Fase d'execució per instruccions de recuperació de pila.**
**b) Diagrama temporal**

### A2.10.3 Instrucció SAVE: *Save Status Register to Stack*

- Format de la instrucció: **SAVE**
- Operació: Guarda en pila, en la posició apuntada per l'apuntador de pila, el contingut del registre d'estat. Després decrementa l'apuntador a pila. Són dues operacions:

  Pila ← Registre d'estat
  SP ← SP-1
- Canvis en registre d'estat: no el modifica.
- La figura A2.19 mostra l'execució de la instrucció. La cronologia és semblant a la de la instrucció PUSH.
- Recordar que cal guardar el registre d'estat sempre que s'executi una rutina d'interrupció que el pugui modificar.

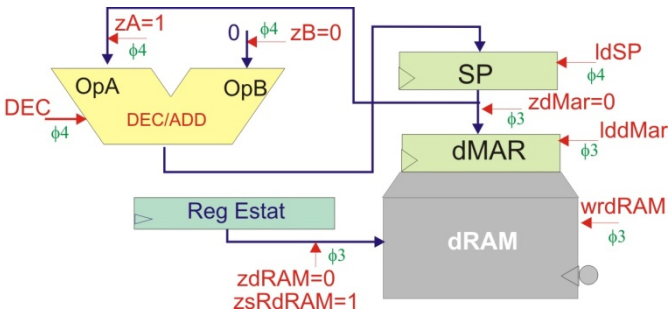


**Figura A2.19. Fase d'execució per instruccions de recuperació de pila.**

### A2.10.4 Instrucció RESTORE: *Restore from Stack*

- Format de la instrucció: **RESTORE**
- Operació: Primer incrementa l'apuntador a pila i després recupera el registre d'estat. Són dues operacions:

  SP ← SP+1
  Registre d'estat ← Pila
- Canvis en registre d'estat: no el modifica.

Apèndix A2. Cicle d'instrucció del Joc d'instruccions del processador EduP12.

- La figura A2.20 mostra l'execució de la instrucció. La cronologia és semblant a la de la instrucció POP.
- Recordar que s'ha de recuperar el registre d'estat sempre que s'hagi executat una rutina d'interrupció en la que s'hagi guardat prèviament.

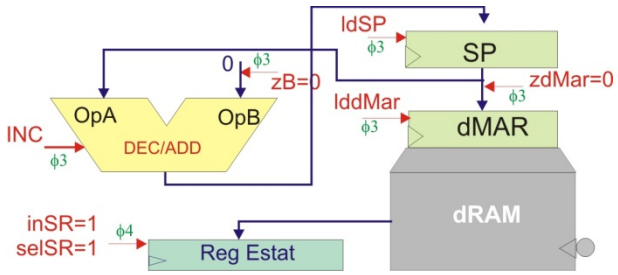


**Figura A2.20. Fase d'execució per instruccions de recuperació de pila.**

### A2.10.5 Instrucció NOP: *No Operation*

- Format de la instrucció: **NOP**
- Operació: No fer res.

  Permet fer temps d'espera sense fer cap operació
- Canvis en registre d'estat: no el modifica.

### Fase d'execució

La fase d'execució és no fer res. Deixa passar, per tant, un cicle de rellotge.