

# Laboratorul 10 - Calculatorul Didactic:

## Adresarea indirectă

Scopul laboratorului este de a completa instrucțiunile implementate până acum cu noi moduri de adresare, și anume: adresarea indirectă prin registru și adresarea indirectă prin sumă de registre. După cum știm, specificarea modului de adresare a unui operand se face în câmpul *MOD* din codificarea instrucțiunii. Modurile de adresare pe care le vom implementa în cadrul acestui laborator sunt identificate prin valoarea *00* a câmpului *MOD*.

Ca și în cazul laboratorului anterior, veți avea de modificat modulul unității de comandă pentru a introduce decodificarea și comandarea prin semnale a execuției instrucțiunilor cu modurile de adresare menționate mai sus.

Detaliile referitoare la componentele calculatorului didactic, codificarea instrucțiunilor (registru RI) și modurile de adresare le găsiți în [curs](#) și în [cheat sheet](#).

## Moduri de adresare

Procesorul calculatorului didactic suportă mai multe moduri de adresare a operanzilor. Până acum am tratat doar cazul operanzilor aflați în registre (adresare directă la registru), identificat prin câmpul *MOD* având valoarea *11*. Cu acest mod de adresare suntem însă limitați la a lucra numai cu cele 8 registre generale. Pentru a putea lucra și cu date aflate în memoria RAM, procesorul calculatorului didactic oferă o serie de alte moduri de încărcare a operanzilor.

Încărcarea unui operand din memoria RAM necesită determinarea adresei la care se află stocat acel operand. Aceasta poate fi stocată direct în instrucțiune, cum se întâmplă în cazul adresării directe, unde instrucțiunea este formată din 2 cuvinte (pe 16 biți), iar al doilea cuvânt conține valoarea adresei (numită și deplasament), sau poate fi calculată folosind diferite combinații de registre cu/fără deplasament. Calculul se face prin adunarea valorilor curente ale unor registre (nu toate combinațiile sunt suportate) și folosirea valorii rezultate ca o adresă pentru memorie.

## Adresarea indirectă prin registru

Adresa efectivă se găsește într-unul din registrele *XA*, *XB*, *BA* sau *BB*. Modul de determinare a acesteia este prezentat în [Fig. 1](#).

Ex: `mov RA, [BA]`

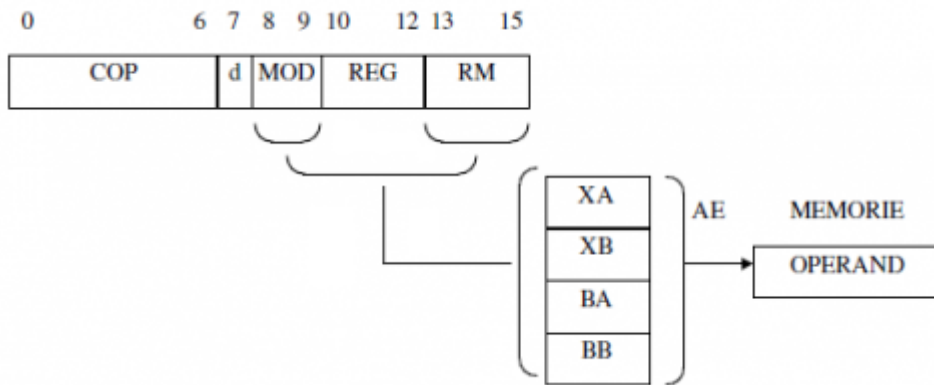


Fig. 1: Adresare indirectă prin registru

## Adresarea indirectă prin sumă de registre

Adresa efectivă se obține prin sumarea conținutului unui registru de bază (*BA* sau *BB*) cu conținutul unui registru index (*XA* sau *XB*). Modul de determinare a acesteia este prezentat în Fig. 2.

Ex: `mov RA, [BA + XA]`

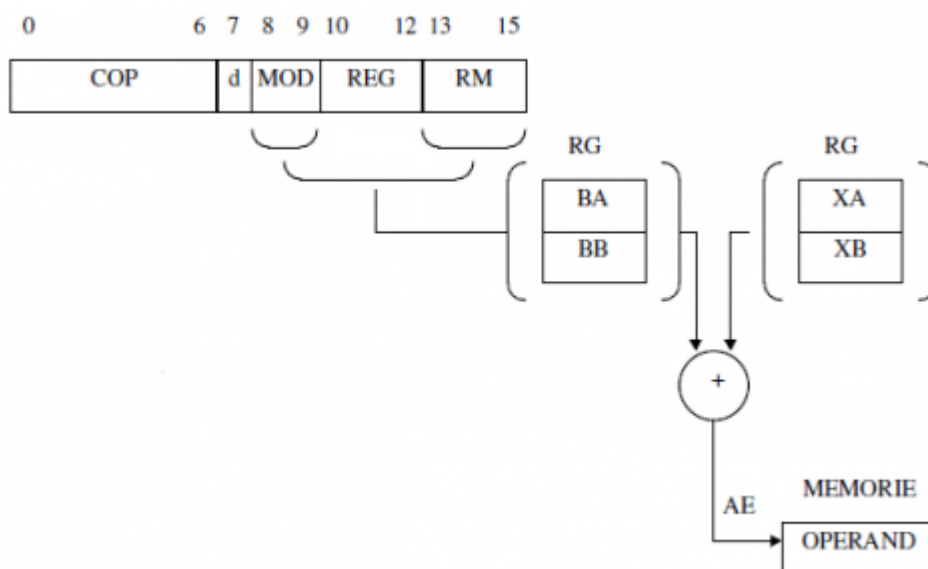


Fig. 2: Adresare indirectă prin sumă de registre

## Implementare

Automatul care trebuie implementat în UC este descris în diagrama de stări din Fig. 3.

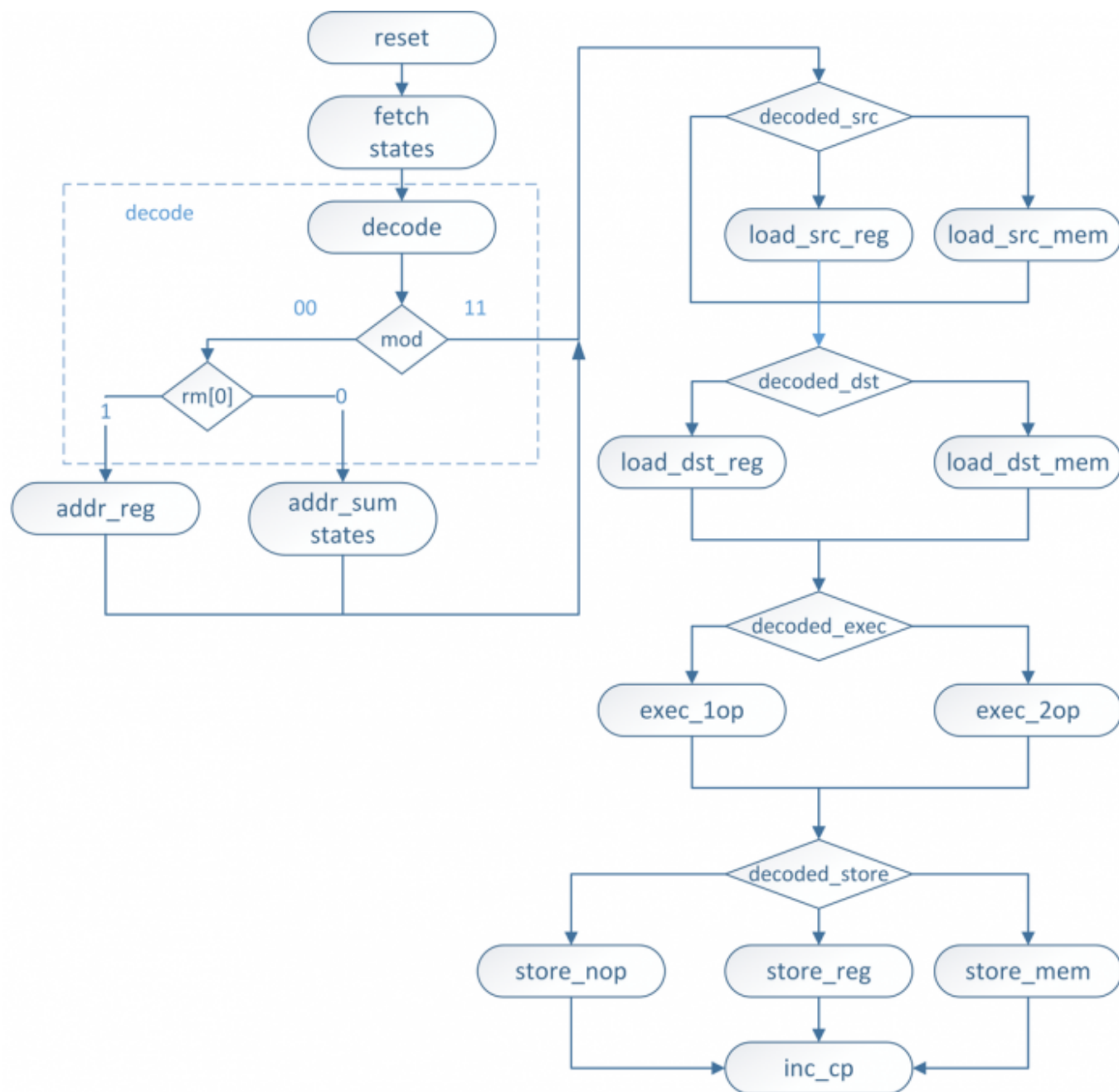


Fig. 3: Diagrama de stări a unității de comandă

După cum puteți observa, la stările deja implementate în laboratoarele anterioare au fost adăugate o serie de stări noi, necesare pentru adresarea indirectă prin registru sau sumă de registre. În cadrul acestui laborator va trebui să implementați aceste noi stări și să completați condițiile corecte pentru decodificarea noilor moduri de adresare.

#### Modificarea conținutului memoriei RAM a calculatorului didactic

Memoria calculatorului didactic este implementată cu ajutorul unui IP Core Xilinx de tipul Block Memory. Parametrii unui core sunt stocați într-un fișier .xco. Acest fișier este folosit de utilitarul de generare a IP Core-urilor pentru a genera o instanță a core-ului selectat. Core-ul de tip Block Memory oferă o serie de parametri care pot fi configurați, precum dimensiunea cuvântului, numărul de cuvinte și semnalele disponibile. Block Memory oferă și posibilitatea inițializării memoriei în momentul generării. Conținutul memoriei poate fi specificat într-un fișier .coe referențiat apoi de fișierul .xco. În figura de mai jos puteți observa formatul fișierul .coe. Valorile din fișier sunt mapate în memoria generată începând de la adresa 0.

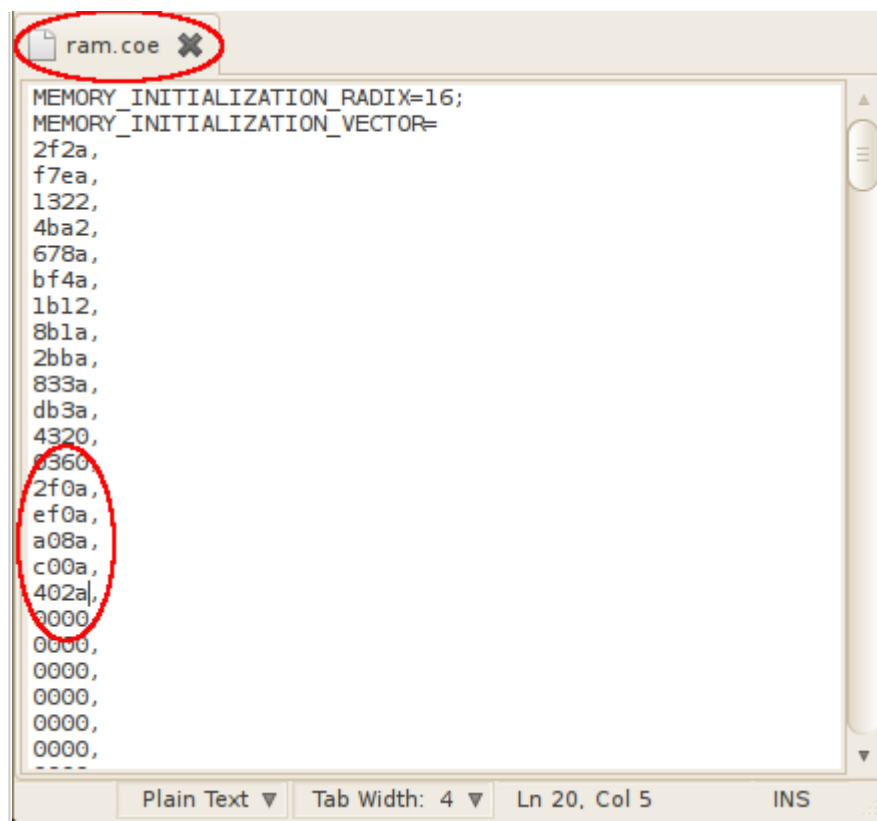


Fig. 4: Fisierul ram.coe

Conținutul inițial al memoriei calculatorului didactic se găsește în cadrul acestui fișier. După fiecare modificare a fișierului .coe, memoria care folosește acel fișier trebuie regenerată. În cazul nostru, după modificarea fișierului *ram.coe* trebuie regenerat modulul *ram* pentru ca modificările să aibă efect. Pentru aceasta, se selectează din fereastra *Sources* modulul *ram*, iar din fereastra *Processes* se inițiază regenerarea, cu opțiunea *Regenerate Core*.

## Exerciții

1. **(4p)** Încărcarea operandului sursă folosind modul de adresare indirectă prin registru.
  1. Adăugați starea *addr\_reg* de determinare a adresei în cazul adresării indirecte prin registru.
    - Hint: Stocați adresa determinată în registrul temporar corespunzător operandului (operandul destinație → *T1*, operandul sursă → *T2*).
  2. Adăugați stările *load\_src\_mem* de încărcare a operandului sursă din memorie.
    - Hint: Adresa operandului a fost calculată și se află în registrul temporar corespunzător operandului sursă (*T2*).
    - Hint: Urmăriți stările *fetch* pentru un exemplu de citire a unei valori din memorie.
  3. Adăugați condițiile de decodificare a modului de calcul al adresei și a locației operandului sursă
  4. După implementarea corectă se vor executa instrucțiunile 1-13 din test.
2. **(2p)** Încărcarea operandului sursă folosind modul de adresare indirectă prin sumă de registre.
  1. Adăugați stările *addr\_sum* de calculare a adresei în cazul adresării indirecte prin sumă de registre.
    - Hint: Stocați adresa calculată în registrul temporar corespunzător operandului (operandul destinație → *T1*, operandul sursă → *T2*).
  2. După implementarea corectă se vor executa instrucțiunile 1-15 din test.
3. **(2p)** Încărcarea operandului destinație folosind cele două moduri de adresare.
  1. Adăugați stările *load\_dst\_mem* de încărcare a operandului destinație din memorie.

- Hint: Adresa operandului a fost calculată și se află în registrul temporar corespunzător operandului destinație (*T1*).
- 2. Adăugați condițiile de decodificare a locației operandului destinație.
- 3. După implementarea corectă se vor executa instrucțiunile 1-18 din test.
- 4. **(2p)** Salvarea rezultatului folosind cele două moduri de adresare.
  - 1. Adăugați stările *store\_mem* de stocare a rezultatului în memorie.
    - Hint: Adresa destinației se află încă stocată în registrul *AM*. De ce?
    - Hint: Memoria are nevoie de un ciclu de ceas pentru a face scrierea după ce a primit adresa, valoarea și semnalul de scriere. Introduceți o întârziere de 1 ciclu.
  - 2. Adăugați condițiile de decodificare a locației rezultatului.
  - 3. După implementarea corectă se vor executa toate instrucțiunile (40) din test.
- 5. **(1p)** Testați corectitudinea implementării pe FPGA.
  - Hint: Urmăriți comentariile din programul de test (aflat în *tests/mod\_00/test.asm*) pentru a vedea rezultatele așteptate.

## Resurse

- [Schelet de cod](#)
- [Soluție laborator](#) (disponibilă începând cu 7.12.2019)
- [PDF laborator](#)
- [Cheat-sheet calculator didactic](#)
- [Arhitectura calculatorului didactic](#)

From:

<https://elf.cs.pub.ro/ac/wiki/> - **AC Wiki**

Permanent link:

<https://elf.cs.pub.ro/ac/wiki/lab/lab10>

Last update: **2019/12/01 15:09**

