Laboratorul 8 - Calculatorul Didactic: Instrucțiuni cu un operand

Obiectivul acestui laborator și al celor care urmează după el, îl reprezintă familiarizarea cu **formatul instrucțiunilor** calculatorului didactic și cu **modul de funcționare al unității de comandă**. În acest scop se vor implementa în Verilog **interpretarea** și **comandarea execuției** pentru instrucțiunile specificate în arhitectura calculatorului didactic studiat la curs.

În laboratorul curent vom implementa unitatea de comandă pentru instrucțiunile aritmetice și logice cu un singur operand.

Unitatea de comandă

Componentele calculatorului didactic implementate în laboratoarele precedente (registre, UAL) formează unitatea de execuție a procesorului. Pentru ca acestea să rețină date și să execute instrucțiunile procesorului, avem nevoie de o logică hardware de comandă a acestora, logică implementată în unitatea de comandă.

În interiorul unui procesor, instrucțiunile trec prin mai multe etape. Pentru calculatorul didactic avem următoarele faze:

- *Fetch* - aducerea instrucțiunii din memorie în registrul instrucțiune (RI)
- *Decode* - decodificarea instrucțiunii
- *Load* - încărcarea operanzilor
- *Execute* - executarea instrucțiunii
- *Store* - scrierea rezultatului (dacă este cazul)

Unitatea de comandă este implicată în toate etapele de mai sus: comandă prin semnalare aducerea codurilor instrucțiunilor din memorie, le decodifică și transmite semnale către unitățile (registre, ual, memorie) implicate în execuția acelor instrucțiuni. La terminarea execuției fiecărei instrucțiuni se comandă scrierea rezultatului (dacă este cazul) și se actualizează registrul *CP* (Contor Program) cu adresa instrucțiunii următoare.

Ce trebuie să facă unitatea de comandă atunci când trebuie executată o instrucțiune aritmetică logică cu un operand? De exemplu `INC RA`.

1. Semnale către *CP* și *AM*: valoarea din *CP* e pusă pe magistrală și scrisă în *AM*;
2. Semnale către *AM* și memorie pentru a lua valoarea (codul instrucțiunii) de la adresa specificată de *AM*;
3. Semnal către *RAM* pentru a pune valoarea pe magistrală și către *RI* pentru a pune codul instrucțiunii în el;
4. Decodificare instrucțiune;
5. Semnal către blocul de registre generale pentru a determina activarea conținutului lui RA pe

magistrală și semnal către *T1* pentru a încărca valoarea aflata în acest moment pe magistrală;
6. Semnal către *UAL* ce indică operația *INC*;
7. Semnal către *IND* pentru ca UAL-ul să poată scrie flagurile;
8. Semnal către blocul de registre generale pentru a încărca în *RA* valoarea de pe magistrală;
9. Incrementarea *CP* pentru adresa instrucțiunii următoare - semnal către *CP* pentru a scrie în el.

Implementare

Unitatea de comandă este implementată ca un **automat de stări**. Modulul acesteia are următoarele semnale:

- intrări: *clk*, *rst*, *ri* (codul instrucțiunii), *ind* (indicatorii de condiție)
- ieșiri:
  - semnale *oe* (output-enable) și *we* (write-enable) pentru registre, bancul de registre, memorie și unitatea aritmetică logică (doar *oe*)
  - *alu_opcode* - codul operației ce trebuie efectuată de unitatea aritmetică logică
  - *alu_carry* - carry-ul folosit de UAL în cadrul operației ce trebuie să o execute
  - *regs_addr* - indexul unui registru din bancul de registre
  - *ind_sel* - controlează sursa de scriere în registrul IND (0 = bus, 1 = alu flags)

Automatul trebuie să ofere stări pentru:

- aducerea instrucțiunii din memorie în registrul *RI*
- decodificarea codului instrucțiunii pentru identificarea operației ce trebuie efectuate și a operanzilor acesteia (dacă este cazul)
- interpretarea fiecărei instrucțiuni. Aceasta se traduce printr-o serie de stări care setează semnalele de output ale modulului pentru a comanda execuția instrucțiunii.
- incrementarea registrului *CP*

În implementarea unității de comandă vom considera că UAL-ul va pune rezultatul în T1, și de acolo va fi transferat în registre sau în memorie. Această convenție face mai simple și mai clare stările care comandă execuția operațiilor aritmetice-logice.

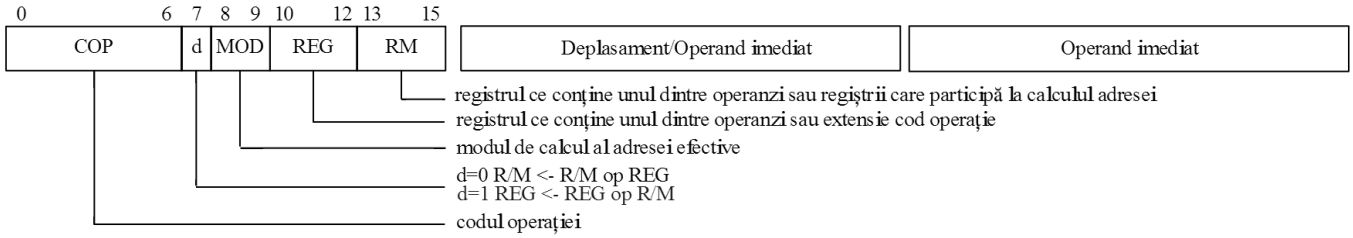
Codificarea instrucțiunilor

Instrucțiune	Funcție	Cod RI[0:6]
INC	op = op + 1	0001 000
DEC	op = op - 1	0001 001
NEG	op = -op	0001 010
NOT	op = ~op	0001 011
SHL/SAL	op = op << 1	0001 100
SHR	op = op >> 1	0001 101
SAR	op = op >>> 1	0001 110

Instrucțiuni aritmetice-logice cu un operand

De exemplu, pentru instrucțiunea `INC RA` grupul este cel al operațiilor cu calcul de adresă efectivă (RI[0] = 0), cu un singur operand (RI[1] = 0), fără operand imediat (RI[2] = 0) și cu salvarea rezultatului (RI[3] = 1).

Pentru orice procesor, fiecare instrucțiune definită în arhitectura setului său de instrucțiuni, are un anumit *cod unic după care este identificată*. Atunci când se stochează o instrucțiune care lucrează cu cel puțin un operand, nu este suficient să avem doar codul său (care indică ce acțiune trebuie efectuată), ci trebuie să avem și niște biți care să ne indice de unde luăm operanzii, așa cum este ilustrat și în imaginea de mai jos.



Formatul instrucțiunilor calculatorului didactic

- *COP* - codul operației, 7 biți
  - bitul [0] - separă instrucțiunile care folosesc o adresă efectivă de cele care nu folosesc:
    - 0 - instrucțiuni cu calcul de adresă efectivă
    - 1 - instrucțiuni fără calcul de adresă efectivă (salturi condiționate, RET etc.)
  - bitul [1] - separă instrucțiunile care au 1 operand de cele cu 2 operanzi:
    - 0 - un operand
    - 1 - doi operanzi
  - bitul [2] - separă instrucțiunile cu operand imediat de cele fără operand imediat:
    - 0 - fără operand imediat
    - 1 - cu operand imediat
  - bitul [3] - separă instrucțiunile de transfer de date/control de celelalte:
    - 0 - transfer de date/control (MOV, PUSH, CALL etc.) sau care nu salvează rezultatul (CMP, TEST)
    - 1 - instrucțiuni aritmetico-logice cu salvarea rezultatului, salturi condiționate
  - biții [4][5][6] - cod operație
- *d* - pentru instrucțiunile cu doi operanzi, folosit pentru a ști care e primul și care e al doilea dintre cele două câmpuri REG și RM din codul instrucțiunii:
  - 0 - RM = RM op REG
  - 1 - REG = REG op RM
- *MOD* - modul de calcul al adresei efective (4 moduri - 2 biți)
- *REG* - indexul registrului care conține unul dintre operanzi
- *RM* - indexul registrului care conține unul dintre operanzi

Dacă MOD = 3 (adresare directă la registru) și instrucțiunea are un singur operand, acesta este pus în RM.

Se poate observa că în cadrul unora din grupurile de operații au rămas codificări nefolosite pe biții 4:6. Dacă se extinde setul de instrucțiuni cu noi operații (vedeți în curs), atunci codul acestora poate fi unul din cele nefolosite (atâta timp cât se încadrează în acel grup).

Exemplu de calcul cod operație pentru instrucțiune DEC RB:

- [0] = 0 - cu calcul de adresă efectivă
- [1] = 0 - un operand
- [2] = 0 - fără operand imediat
- [3] = 1 - operație aritmetică cu salvarea rezultatului
- [4,5,6] = [0,0,1] - codul dat operației (stabilit de arhitectură)
- [7] = 0/1 - nu contează, îl putem pune 0 sau 1
- [8,9] = [1,1] - modul de adresare (directă la registru)
- [10,11,12] = [0,0,0] - nu folosim REG, deci nu contează ce punem
- [13,14,15] = [0,0,1] - indexul registrului RB în bancul de registre

15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1	0	0	0	0	0	1	1	0	1	0	0	1	0	0	0

Exerciții

Modificați modulul *uc.v* din scheletul de cod al laboratorului, astfel încât să implementați instrucțiunile din tabelul prezentat anterior. Citiți secțiunea de implementare.

1. **(4p)** Decodificați instrucțiunea *INC RA*.
   1. Identificați grupul de instrucțiuni în care se încadrează operația (inc) și operandul (RA).
   2. Transferați conținutul registrului RA în registrul T1
   3. Indicați UAL-ului să execute operația și puneți rezultatul în T1
      - registrul IND trebuie conectat la UAL ca să poată fi setate flagurile în urma execuției operației
   4. Transferați conținutul registrului T1 în registrul RA
2. **(1p)** Generalizați instrucțiunea implementată la exercițiul anterior astfel încât să funcționeze cu orice registru general.
3. **(5p)** Decodificați restul instrucțiunilor.

Testați întâi corectitudinea implementării folosind testerul offline. Apoi testați pe placă. Ram-ul are încărcate instrucțiunile din *tests→one_operand→test.asm*.

Fișierul *calc_didatic→ram.coe* conține instrucțiunile programului din fișierul de test reprezentate în sistem hexazecimal. Adresa fiecărei instrucțiuni este dată de indicele instrucțiunii respective în vectorul de inițializare al memoriei.

Pentru a schimba programul rulat pe placă este suficient să modificați fișierul *calc_didatic→ram.coe*, urmând ca apoi să dați *yes* la *regenerate core*.

Pentru a schimba programul rulat de către tester modificați fișierul *tests→one_operand→test.coe*.

Interacțiunea cu butoanele de pe placă pentru afișajul pe display-ul LCD:

- se poate controla secțiunea afișată pe LCD cu *BTN_WEST* si *BTN_EAST*, iar în cadrul secțiunii puteți face scroll cu butonul rotativ;
- clk manual pe *BTN_SOUTH*, acesta este activat cand toate switch-urile sunt în jos;
- resetul sincron pe *BTN_NORTH*;
- switch-urile controlează factorul de divizare pentru generatorul de ceas de pe placă. Toate switch-urile în sus corespund factorului cel mai mare

Resurse

- Schelet de cod
- PDF laborator
- Cheat-sheet calculator didactic
- Arhitectura calculatorului didactic
- Soluție laborator