**Universitatea Politehnica Timisoara**

**Facultatea de Automatica si Calculatoare**

**Programul de studii: Calculatoare si Tehnologia Informatiei**

**Calculator de buzunar**

**General Purpose processor**

**Fundamente de ingineria calculatoare**

**Echipa *CalcCrafters***

**Project Manager:** Spatacean Oana- Teodora

**Hardware Design:** Stanescu Ariana- Camelia, Tarod Anamaria- Alexandra

**Software Design:** Oxani Corina

**Testare:** Talapan Alexandra

**Repository link pe Github:** <https://github.com/OanaSpatacean/Calculator_de_buzunar>

# **Cuprins**

## **1 Tema proiectului** **2 Setul de instructiuni si formatele de instructiuni**

Instructiunile de branch

Instructiuni aritmetice si logice

Instructiune care muta valoarea immediate intr-un registru

Instructiuni care lucreaza cu memoria

Instructiuni de copiere

Instructiuni care folosesc stiva

Instructiune care nu realizeaza nicio operatie

## **3 Design hardware**

Schema generatorului de semnale de control (Control Signal Generator)

Intrarile generatorului

Iesirile generatorului

Intrari

Flag-uri

## **4 Design software**

## **5 Testare**

## **6 Bibliografie**

**1.**  **TEMA PROIECTULUI**

Proiectul nostru are ca temă centrală dezvoltarea unui procesor de uz general care să permită efectuarea operațiilor întâlnite într-un calculator de buzunar obișnuit.

Un procesor de uz general care este o unitate centrală de prelucrare care nu este proiectată să servească un scop specific. Acesta poate fi utilizat într-o varietate de aplicații și poate fi adaptat pentru a satisface nevoile unei anumite utilizări.

**2. SETUL DE INSTRUCTIUNI SI FORMATELE DE INSTRUCTIUNI**

Avem 3 registrii ACC, X, Y

Codificare: X – 0 ; Y – 1

**Instructiunile de branch:**

|  |  |  |
| --- | --- | --- |
| Nume instructiune – Opcode <6> | Address <10> | Descriere |
| BRZ – 000000 | <10> | Se actualizeaza PC cu adresa de branch specificata de cei mai nesemnificativi 10 biti doar daca flag-ul Zero(Z) este activ. |
| BRN – 000001 | <10> | Se actualizeaza PC cu adresa de branch specificata de cei mai nesemnificativi 10 biti doar daca flag-ul Negative(N) este activ. |
| BRC – 000010 | <10> | Se actualizeaza PC cu adresa de branch specificata de cei mai nesemnificativi 10 biti doar daca flag-ul Carry(C) este activ. |
| BRO – 000011 | <10> | Se actualizeaza PC cu adresa de branch specificata de cei mai nesemnificativi 10 biti doar daca flag-ul Overflow(V) este activ. |
| BRA – 000100 | <10> | Se actualizeaza PC cu adresa de branch specificata de cei mai nesemnificativi 10 biti. |
| JMP – 000101 | <10> | Se apeleaza procedura aflata la adresa specificata de cei mai nesemnificativi 10 biti. |
| RET – 000110 | <10> | Se revine din procedura executandu-se instructiunea de dupa apelul procedurii. |

**Instructiuni aritmetice si logice:**

|  |  |  |  |
| --- | --- | --- | --- |
| Nume instructiune – Opcode <6> | Address  <1> | Immediate  <9> | Descriere |
| ADD – 000111 | <1> | <9> | Daca Immediate este 0, atunci se realizeaza operatia ACC=ACC+X sau ACC=ACC+Y, in functie de adresa registrului. Daca Immediate este diferit de 0, atunci se realizeaza operatia  X=X+Immediate sau Y=Y+Immediate, in functie de adresa registrului. |
| SUB – 001000 | <1> | <9> | Daca Immediate este 0, atunci se realizeaza operatia ACC=ACC-X sau ACC=ACC-Y, in functie de adresa registrului. Daca Immediate este diferit de 0, atunci se realizeaza operatia  X=X-Immediate sau Y=Y-Immediate, in functie de adresa registrului. |
| LSR – 001001 | <1> | <9> | Daca Immediate este 0, atunci se realizeaza operatia ACC=ACC>>X sau ACC=ACC>>Y, in functie de adresa registrului. Daca Immediate este diferit de 0, atunci se realizeaza operatia  X=X>>Immediate sau Y=Y>>Immediate, in functie de adresa registrului. |
| LSL – 001010 | <1> | <9> | Daca Immediate este 0, atunci se realizeaza operatia ACC=ACC<<X sau ACC=ACC<<Y, in functie de adresa registrului. Daca Immediate este diferit de 0, atunci se realizeaza operatia  X=X<<Immediate sau Y=Y<<Immediate, in functie de adresa registrului. |
| RSR – 001011 | <1> | <9> | Daca Immediate este 0, atunci se realizeaza operatia de rotire la dreapta al ACC cu atatea pozitii cu cat este specificat in X sau Y, in functie de adresa registrului. Daca Immediate este diferit de 0, atunci se realizeaza operatia de rotire la dreapta cu valoarea Immediate a lui X sau Y, in functie de adresa registrului. |
| RSL – 001100 | <1> | <9> | Daca Immediate este 0, atunci se realizeaza operatia de rotire la stanga al ACC cu atatea pozitii cu cat este specificat in X sau Y, in functie de adresa registrului. Daca Immediate este diferit de 0, atunci se realizeaza operatia de rotire la stanga cu valoarea Immediate a lui X sau Y, in functie de adresa registrului. |
| MUL – 001101 | <1> | <9> | Daca Immediate este 0, atunci se realizeaza operatia ACC=ACC\*X sau ACC=ACC\*Y, in functie de adresa registrului. Daca Immediate este diferit de 0, atunci se realizeaza operatia  X=X\*Immediate sau Y=Y\*Immediate, in functie de adresa registrului. |
| DIV – 001110 | <1> | <9> | Daca Immediate este 0, atunci se realizeaza operatia ACC=ACC/X sau ACC=ACC/Y, in functie de adresa registrului. Daca Immediate este diferit de 0, atunci se realizeaza operatia  X=X/Immediate sau Y=Y/Immediate, in functie de adresa registrului. |
| MOD - 001111 | <1> | <9> | Daca Immediate este 0, atunci se realizeaza operatia ACC=ACC%X sau ACC=ACC%Y, in functie de adresa registrului. Daca Immediate este diferit de 0, atunci se realizeaza operatia  X=X%Immediate sau Y=Y%Immediate, in functie de adresa registrului. |
| OR – 010000 | <1> | <9> | Daca Immediate este 0, atunci se realizeaza operatia ACC=ACC|X sau ACC=ACC|Y, in functie de adresa registrului. Daca Immediate este diferit de 0, atunci se realizeaza operatia  X=X|Immediate sau Y=Y|Immediate, in functie de adresa registrului. |
| AND – 010001 | <1> | <9> | Daca Immediate este 0, atunci se realizeaza operatia ACC=ACC&X sau ACC=ACC&Y, in functie de adresa registrului. Daca Immediate este diferit de 0, atunci se realizeaza operatia  X=X&Immediate sau Y=Y&Immediate, in functie de adresa registrului. |
| XOR – 010001 | <1> | <9> | Daca Immediate este 0, atunci se realizeaza operatia ACC=ACC^X sau ACC=ACC^Y, in functie de adresa registrului. Daca Immediate este diferit de 0, atunci se realizeaza operatia  X=X^Immediate sau Y=Y^Immediate, in functie de adresa registrului. |
| NOT – 010011 | <1> | 000000000 | Se inverseaza bitii din X sau Y in functie de adresa registrului |
| CMP – 010100 | <1> | <9> | Daca Immediate este 0, atunci se compara ACC cu X sau cu Y, in functie de adresa registrului. Daca Immediate este diferit de 0, atunci se compara X sau Y cu Immediate, in functie de adresa registrului. Compararea se realizeaza prin scaderea celor 2 valori comparate. |
| TST - 010101 | <1> | <9> | Daca Immediate este 0, atunci se compara ACC cu X sau cu Y, in functie de adresa registrului. Daca Immediate este diferit de 0, atunci se compara X sau Y cu Immediate, in functie de adresa registrului. Compararea se realizeaza prin operatia SI pe biti intre cele 2 valori. |
| INC – 010110 | <1> | 000000000 | Se incrementeaza valoarea lui X sau Y, in functie de adresa. |
| DEC – 010111 | <1> | 000000000 | Se decrementeaza valoarea lui X sau Y, in functie de adresa. |
| FACT – 011000 | <1> | <9> | Se realizeaza operatia factorial a valorii din Immediate si rezultatul este plasat in X sau Y, in functie de adresa specificata. |
| PATRAT – 011001 | <1> | <9> | Se realizeaza operatia de ridicare la patrat a valorii din Immediate si rezultatul este plasat in X sau Y, in functie de adresa specificata. |

**Instructiune care muta valoarea immediate intr-un registru:**

|  |  |  |  |
| --- | --- | --- | --- |
| Nume instructiune – Opcode <6> | Address  <1> | Immediate  <9> | Descriere |
| MOV – 011010 | <1> | <9> | Muta valoarea data de Immediate in registrul X sau Y, in functie de valoarea registrului. |

**Instructiuni care lucreaza cu memoria:**

|  |  |  |  |
| --- | --- | --- | --- |
| Nume instructiune – Opcode <6> | Address  <1> | Immediate  <9> | Descriere |
| LD – 011011 | <1> | <9> | Incarca in X sau Y in functie de valoarea specificata de adresa, valoarea din memorie specificata de Immediate |
| ST - 011100 | <1> | <9> | Incarca in memorie la adresa specificata de Immediate valoarea din X sau Y in functie de valoarea specificata de adresa. |

**Instructiuni de copiere:**

|  |  |  |
| --- | --- | --- |
| Nume instructiune – Opcode <6> | Unused  <1> | Descriere |
| COPY\_X – 011101 | 0000000000 | Copiaza valoarea din ACC in X. |
| COPY\_Y - 011110 | 0000000000 | Copiaza valoarea din ACC in Y. |

**Instructiuni care folosesc stiva:**

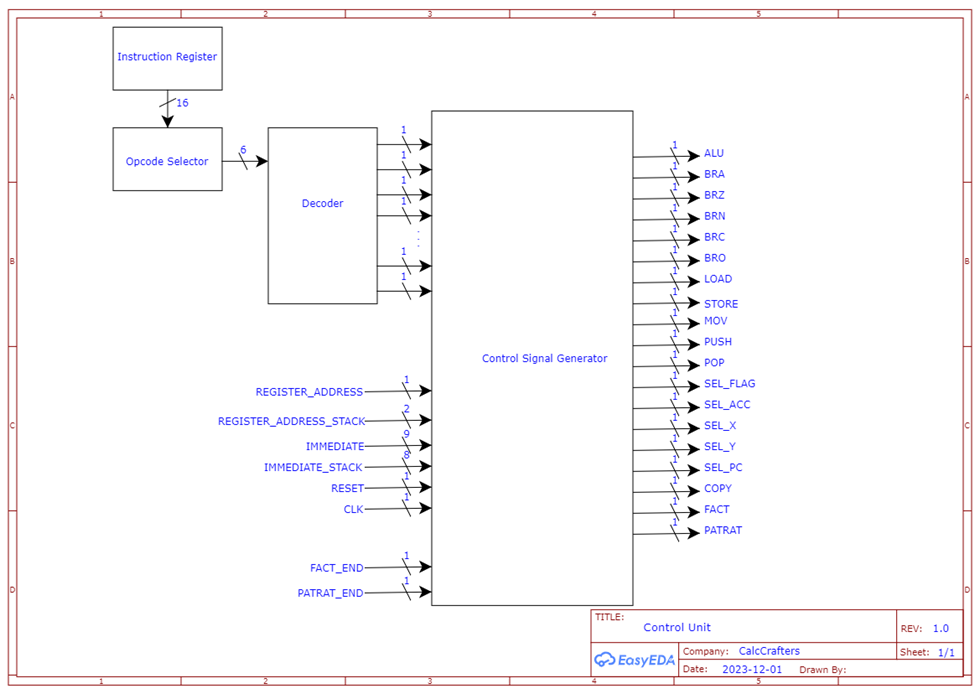
|  |  |  |  |
| --- | --- | --- | --- |
| Nume instructiune – Opcode <6> | Address  <1> | Immediate  <8> | Descriere |
| PUSH – 011111 | <2> | 00000000 | Incarca in memorie la adresa specificata de Stack Pointer valoarea din ACC, X sau Y, in functie de adresa. |
| POP - 100000 | <2> | 00000000 | Incarca valoarea din memorie de la adresa specificata de Stack Pointer in ACC, X sau Y, in functie de adresa. |

**Instructiune care nu realizeaza nicio operatie:**

|  |  |  |
| --- | --- | --- |
| Nume instructiune – Opcode <6> | Unused  <1> | Descriere |
| NOP – 100001 | 0000000000 | Genereaza un delay de un ciclu de clock. |

**3. DESIGN HARDWARE**

**SCHEMA GENERATORULUI DE SEMNALE DE CONTROL (CONTROL SIGNAL GENERATOR):**



**INTRARILE GENERATORULUI**:

**OPCODE** care este decodificata cu ajutorul unui decodificator

**REGISTER**\_**ADDRESS** intrare de decizie folosita pentru instructiunile in care adresa este pe un bit (cand se folosesc registrii X si Y)

**REGISTER**\_**ADDRESS**\_**STACK** intrare de decizie folosita pentru instructiunile de PUSH si POP, pentru care adresa este pe 2 biti (cand se folosesc registrii ACC, X si Y)

**IMMEDIATE** intrare de decizie pe 9 biti folosita cand adresa este pe un bit

**IMMEDIATE**\_**STACK** intrare de decizie pe 8 biti folosita pentru instructiunile PUSH si POP

**CLK**

**RESET**

**FACT**\_**END** intrare care dezactiveaza modulul responsabil de calculul factorialului

**PATRAT**\_**END** intrare care dezactiveaza modulul responsabil de calculul patratului unui numar

**IESIRILE GENERATORULUI**

**ALU**: se pune pe 1 cand se executa o instructiune aritmetica sau logica – intrarea de enable de la ALU

**BRA**: se pune pe 1 atunci cand se realizeaza o instructiune de salt (conditionat sau neconditionat)

**BRZ**: se pune pe 1 atunci cand se executa o instructiune de salt conditonat cu flag-ul ZERO

**BRN**: se pune pe 1 atunci cand se executa o instructiune de salt conditonat cu flag-ul NEGATIVE

**BRC**: se pune pe 1 atunci cand se executa o instructiune de salt conditonat cu flag-ul CARRY

**BRO**: se pune pe 1 atunci cand se executa o instructiune de salt conditonat cu flag-ul OVERFLOW

**LOAD**: se pune pe 1 cand se preiau date din Data Memory (inclusive la POP)

**STORE**: se pune pe 1 cand se incarca date in Data Memory (inclusiv la PUSH)

**MOV**: se pune pe 1 atunci cand se executa o instructiune de mutare a valorii immediate in X sau Y

**PUSH**: se pune pe 1 atunci cand se executa instructiunea de PUSH

**POP**: se pune pe 1 atunci cand se executa instructiunea de POP

**SEL\_FLAG**: se pune pe 1 atunci cand se modifica valorile din registrul de flags (doar in cazul operatiilor aritmetice si logice)

**SEL\_ACC**: se pune pe 1 atunci cand se actualizeaza acumulatorul (S=0) sau se preiau valori din accumulator pentru a fi incarcate in Data Memory (S=1). Pentru instructiunile aritmetice si logice se pune pe 1 doar cand IMMEDIATE=0

**SEL\_X**: se pune pe 1 atunci cand se actualizeaza X sau se preiau valori din X pentru a fi incarcate in Data Memory. Pentru instructiunile aritmetice si logice se pune pe 1 doar cand IMMEDIATE!=0 si REGISTER\_ADDRESS=0

**SEL\_Y**: se pune pe 1 atunci cand se actualizeaza Y sau se preiau valori din Y pentru a fi incarcate in Data Memory. Pentru instructiunile aritmetice si logice se pune pe 1 doar cand IMMEDIATE!=0 si REGISTER\_ADDRESS=1

**SEL\_PC**: se pune pe 1 atunci cand se actualizeaza PC sau cand se preiau valori din PC pentru a fi incarcate in Data Memory

**COPY**: se pune pe 1 atunci cand se realizeaza o instructiune de copiere din ACC in X sau Y

**FACT**: se pune pe 1 atunci cand se executa instructiunea factorial

**PATRAT**: se pune pe 1 atunci cand se executa instructiunea de ridicare la patrat

Pentru semnalele SEL\_ACC, SEL\_X, SEL\_Y, SEL\_PC, pentru actualizarea registrului S=0, iar pentru preluarea datelor S=1 (pentru multiplexorul de selectie a intrarii in Data Memory).

**ALU**

**INTRARI:**

**ACC**

**X**

**Y**

**IMMEDIATE**

**ALU** contine o unitate de control locala care selecteaza operatiile aritmetice si logice pe baza opcode-ului. Registrii de intrare, cu ajutorul carora se realizeaza operatiile sunt selectati cu ajutorul semnalelor REGISTER\_ADDRESS si IMMEDIATE.

Exemplu: instructiunea ADD

Atunci cand IMMEDIATE are valoarea 0, se va alege utilizarea registrilor ACC si X sau Y.

=> ACC=ACC+X

Atunci cand IMMEDIATE este diferit de 0, se va alege utilizarea registrilor X sau Y si valoarea din IMMEDIATE

=> X=X+IMMEDIATE

Valoarea lui REGISTER\_ADDRESS determina care dintre registrii X si Y va fi ales.

Atunci cand REGISTER\_ADDRESS este 0 va fi ales registrul X.

Atunci cand REGISTER\_ADDRESS este 1 va fi ales registrul Y.

**FLAG-urile** sunt modificate de instructiunile aritmetice si logice.

• **ZERO**: daca in urma efectuarii operatiei aritmetice sau logice rezultatul este 0, acest flag este setat pe 1;

• **NEGATIVE**: daca in urma efectuarii operatiei aritmetice sau logice rezultatul are bitul de semn 1, acest flag este setat pe 1;

• **CARRY**: daca se aduna doua numere iar rezultatul depaseste valoarea maxima alocata pe acel numar de biti), acest flag este setat pe 1. Acest flag nu se seteaza in cazul operatiilor de siftare si impartire, ci ramane pe 0;

• **OVERFLOW**: daca se aduna doua numere de acelasi semn si rezultatul e de semn contrar, acest flag este setat pe 1. Acest flag nu se seteaza in cazul operatiilor de siftare si impartire, ci ramane pe 0;