**16-bit Processor**

**1)Introducere**

Am ales ca tema pentru acest proiect, sa implementam in Verilog un processor pe 16 biti. In partea de jos a paginii, este prezentata schema hardware, unde se pot observa modulele din care se compune:

* *Control Unit (CU) - Unitatea de control* - In acest modul are loc resetarea si incrementarea program counter-ului (PC), dar si a semnalelor ce vor controla celelalte module care sunt implementate pentru procesor.
* *Instruction Memory (IM) – Memoria de instructiuni* – In acest modul vom stoca instructiunile ce se vor executa la simulare (programul rulat de processor).
* *Registers (Data Memory) – Memoria de date* – In acest modul se stocheaza datele de care are nevoie procesorul sa lucreze, mai explicit, memoria interna a acestuia.
* *Arithmetic-Logic Unit (ALU) – Unitatea Aritmetica si Logica* – Aici se vor efectua operatiile propriu-zise.
* *Flags* – Locul unde se semnaleaza diferitele cazuri rezultate in urma unei operatii aritmetice, respectiv zero, negativ, carry, overflow.
* *Stack – Unitatea de memorie* - Modulul de tip stiva unde se pot efectua operatii de tip push si pop.

**2)Unitatea de control**

Unitatea de control -> scopul de a activa semnale pentru a controla celelalte module ale procesorului si de a incrementa Program Counter-ul. In partea de jos se vor prezenta input-urile si output-urile modului, implementat in Verilog.

Semnalul instruction vine din modulul Instruction Memory. Acest semnal este decodificat in Control Unit, de unde vor rezulta 2 semnale:

1. opCode – Operația ce trebuie efectuata
2. immediate – Valoarea care va fi folosita in instructiune.

Semnalele fl\_zero, fl\_negative, fl\_carry, fl\_overflow sunt legate la modulul Flags si ofera detalii despre rezultatul ultimei operatii din ALU. Aceste semnale sunt folosite de instructiunile de salt conditionat.

In ceea ce priveste functionarea CU, aceasta se realizeaza in urmatorul mod: La fiecare front al ciclului de clock, set\_PC ia valoarea lui r\_PC, incrementat cu o unitate, semnaland memoriei de instructiuni faptul ca trebuie sa furnizeze in instruction, urmatoarea instructiune(daca nu este setat PC manual, de o instructiune).

Urmeaza etapa de decode a instructiunii, in urma careia se genereaza cele 2 semnale amintite mai sus, ce vor dicta ceea ce se va intampla in continuare. Pe baza opCode-ului se vor active semnalele de enable ( alu\_en, write\_reg\_en, st\_read\_en, st\_write\_en), dar si semnalele corespunzatoare operatiei (term1, term2 , st\_address, st\_push, set\_ACC, set\_X, set\_Y, set\_SP, set\_LR, set\_PC) care vor comanda celelalte module.

Instructiunea ce vine de la IM (instruction) va avea una din urmatoarele forme: **opcode(6)\_immediate(10)**.

Intrebari:  
- Ar trebui sa nu trecem prin Control Unit si valorile, ci doar sa folosim semnalele de control pentru a le incarca?  
- Sign extension trebuie sa functioneze doar pe 9 biti, tinand cont ca la adrese nu trebuie pastrat niciun semn?  
- Pentru a avea instructiuni care functioneaza si cu registru, si cu immediate ca al doilea parametru, cum decodam instructiunea? Putem sacrifica o valoare de immediate pentru a semnifica faptul ca nu avem immediate si trebuie sa luam din registru?