ALU egység tervezése

# Specifikáció

* A rendszer 8 regiszterrel működik, ebből egy kitüntetett van (AKKU), a többi általános felhasználású.
* 8 bites operandusok vannak, de ebből 1 bit logikailag meg van különböztetve(load bit jelzi, ha adatok töltődik be valamelyik regiszterbe), A felső három bit a regisztert jelöli ki, az alsó négy bit a végrehajtandó művelet kódját jelöli.
* A rendszernek még ezen kívül van *cin*, és *cout* bitje, amik a kaszkádosításhoz, és egyéb műveletekhez kellenek.
* A megvalósított műveletek: ADD, SUB, LSHIFT, RSHIFT, XOR, CMP, AND, NAND, OR, NOR, CPY.
* A shiftelő utasításoknál a kishiftelt bit a cout-ba kerül, a beshiftelt bit pedig cin tartalma.
* Ezen kívül az összeadásnál és kivonásnál is használva van cout és cin értéke, de a többi műveletnél már nem.
* Az egyik operandus mindig az AKKU, a végeredmény mindig az AKKU-ba kerül
* A CPY művelet az opkódban megadott operandus tartalmát belemásolja az AKKU-ba.
* Az ALU köré épített kezdetleges CPU kétféle állapotban lehet: IDLE, és OPERATING
* IDLE állapotban, ha load==1 és ce==1, akkor az opkódban megadott regiszter értékét tölti fel a data\_in buszon lévő értékkel. Ha load==0 és ce==1, akkor a register\_selector értéke felveszi az opkódban megadott regiszter értékét, és a CPU állapota átvált OPERATING állapotba.
* OPERATING állapotban átvált IDLE állapotba, így ha már inicializáltuk a regiszterek értékét, és ce==1, akkor folyamatosan váltogat az OPERATING és IDLE állapot között.

# Blokkvázlat

TODO

# Modulok részletes leírása

TODO

## ALU\_unit.v

Ez tartalmazza a megvalósítandó ALU egységet.

A bemenetei/kimenetei:

* opA[7:0]: az első operandus, a modulon kívül az AKKU van rákötve
* opB[7:0]: a második operandus, a modulon kívül a register\_selector által regiszter kimenete van rákötve
* outData[7:0]: az ALU adatkimenete, a modulon kívül az AKKU bemenetére van rákötve
* opcode[3:0]: Ez határozza meg, hogy melyik műveletet kell elvégeznie az ALU-nak
* cin: bemenet, segítségével a kaszkádosítás, átvitelképzés megvalósítható, valamint a shiftelésnél cin értéke shiftelődik be az opA-ba, a shiftelés irányának megfelelően
* cout:kimenet, segítségével a kaszkádosítás, átvitelképzés megvalósítható, valamint a shiftelésnél a kishiftelt bit értéke a cout-on keresztül adódik tovább.