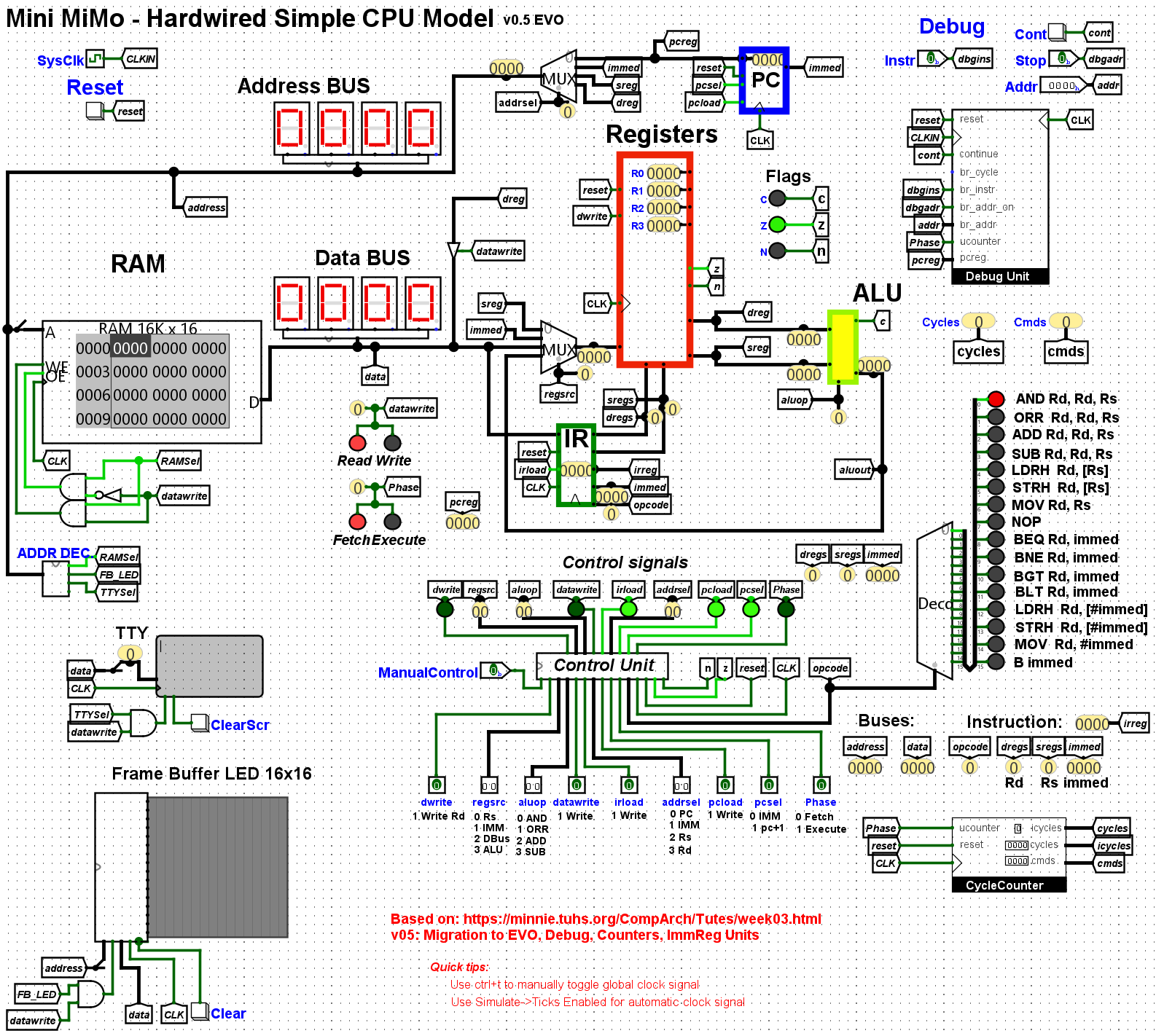
Mini MiMo

****Preprost trdo-ožičen model CPE

v 0.1

# Osnovne značilnosti

* 4 16 bitni registri (R0-R3)
* 16 bitna ALE (+,-,AND,OR)
* 16 ukazov
* 2 izhodni napravi (16x16 zaslon LED in znakovni terminal TTY)
* Preprost zbirnik v Excelu

# Nabor ukazov

## 16 bitni ukazi - format:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| op1 | op2 | Rd | Rs | immediate |
| 2b | 2b | 2b | 2b | 8b |

## Seznam ukazov:

|  |  |  |  |
| --- | --- | --- | --- |
| **op1** | **op2** | **ARM9 zapis** | **Opis** |
| 00 | 00 | AND Rd, Rd, Rs | Rd = Rd AND Rs |
| 00 | 01 | ORR  Rd, Rd, Rs | Rd = Rd OR Rs |
| 00 | 10 | ADD Rd, Rd, Rs | Rd = Rd + Rs |
| 00 | 11 | SUB Rd, Rd, Rs | Rd = Rd - Rs |
| 01 | 00 | LDRH  Rd, [Rs] | Rd = Mem[Rs] |
| 01 | 01 | STRH  Rd, [Rs] | Mem[Rs] = Rd |
| 01 | 10 | MOV Rd, Rs | Rd = Rs |
| 01 | 11 | NOP | Do nothing |
| 10 | 00 | BEQ Rd, immed | PC = immed if Rd == 0 |
| 10 | 01 | BNE Rd, immed | PC = immed if Rd != 0 |
| 10 | 10 | BGTZ Rd, immed | PC = immed if Rd > 0 |
| 10 | 11 | BLTZ Rd, immed | PC = immed if Rd < 0 |
| 11 | 00 | LDRH  Rd, [#immed] | Rd = Mem[immed] |
| 11 | 01 | STRH  Rd, [#immed] | Mem[immed] = Rd |
| 11 | 10 | MOV  Rd, #immed | Rd = immed |
| 11 | 11 | B immed | PC = immed |

Ukazi op1=1X so ukazi s takojšnjim operandom.

Ukazi op1=00 so aritmetično logični ukazi (izvedba v ALE)

# Izvedba ukazov

## FETCH

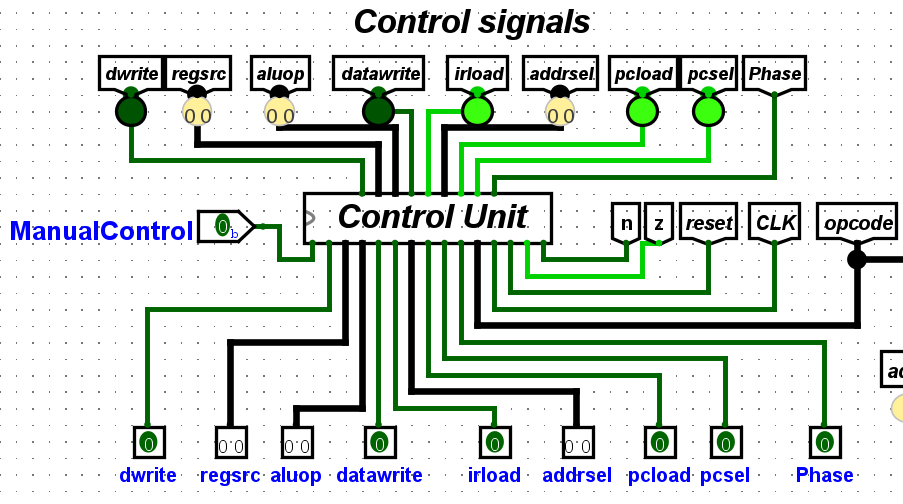
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **op1** | **op2** | **ARM9 zapis** | **pc**  **sel** | **pc**  **load** | **ir**  **load** | **rw** | **d**  **write** | **addr**  **sel** | **reg**  **sel** | **dreg** | **sreg** | **aluop** |
| xx | xx | Vsi ukazi | 1  (pc+1) | 1 | 1 | 0 | 0 | 0  (pc) |  |  |  |  |

## EXECUTE

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **op1** | **op2** | **ARM9 zapis** | **pc**  **sel** | **pc**  **load** | **ir**  **load** | **rw** | **d**  **write** | **addr**  **sel** | **reg**  **sel** | **dreg** | **sreg** | **aluop** |
| 00 | 00 | AND Rd, Rd, Rs |  | 0 | 0 | 0 | 1 |  | 3 | Rd | Rs | op2 |
| 00 | 01 | ORR  Rd, Rd, Rs |  | 0 | 0 | 0 | 1 |  | 3 | Rd | Rs | op2 |
| 00 | 10 | ADD Rd, Rd, Rs |  | 0 | 0 | 0 | 1 |  | 3 | Rd | Rs | op2 |
| 00 | 11 | SUB Rd, Rd, Rs |  | 0 | 0 | 0 | 1 |  | 3 | Rd | Rs | op2 |
| 01 | 00 | LDRH  Rd, [Rs] |  | 0 | 0 | 0 | 1 | 2 | 2 | Rd | Rs |  |
| 01 | 01 | STRH  Rd, [Rs] |  | 0 | 0 | 1 | 0 | 3 |  | Rd | Rs |  |
| 01 | 10 | MOV Rd, Rs |  | 0 | 0 | 0 | 1 |  | 1 | Rd | Rs |  |
| 01 | 11 | NOP |  | 0 | 0 | 0 | 0 |  |  |  |  |  |
| 10 | 00 | BEQ Rd, immed | 0 | j | 0 | 0 | 0 |  |  | Rd |  | op2 |
| 10 | 01 | BNE Rd, immed | 0 | j | 0 | 0 | 0 |  |  | Rd |  | op2 |
| 10 | 10 | BGTZ Rd, immed | 0 | j | 0 | 0 | 0 |  |  | Rd |  | op2 |
| 10 | 11 | BLTZ Rd, immed | 0 | j | 0 | 0 | 0 |  |  | Rd |  | op2 |
| 11 | 00 | LDRH  Rd, [#immed] |  | 0 | 0 | 0 | 1 | 1 | 2 | Rd |  |  |
| 11 | 01 | STRH  Rd, [#immed] |  | 0 | 0 | 1 | 0 | 1 |  | Rd |  |  |
| 11 | 10 | MOV  Rd, #immed |  | 0 | 0 | 0 | 1 |  | 1 | Rd |  |  |
| 11 | 11 | B immed | 0 | 1 | 0 | 0 | 0 |  |  |  |  |  |
| **op1** | **op2** | **ARM9 zapis** | **pc**  **sel** | **pc**  **load** | **ir**  **load** | **rw** | **d**  **write** | **addr**  **sel** | **reg**  **sel** | **dreg** | **sreg** | **aluop** |

V zgornjih tabelah so razvidna stanja vseh krmilnih signalov, ki določajo delovanje sistema v korakih Fetch in Execute za vsak ukaz. V tabelah so navedene le aktivne vrednosti signalov. Če vrednost signala ni pomembna, potem ni navedena.

## Kontrolni signali



Branje (FETCH) in izvedba ukazov (EXECUTE) poteka s pomočjo kontrolnih signalov. Z njimi kontrolna enota krmili delovanje vseh ostalih enot. Z vklopom vhoda »ManualControl« v stanje 1, pa lahko stanje kontrolnih signalov določamo samo s pomočjo vhodov v spodnji vrstici.

Kratek opis kontrolnih signalov :

* **dwrite**
  + 1 .. vpis v register Rd, 0 .. Rd pomni
* **regsrc**
  + izbira vhoda v registrsko enoto
    - 00 .. Rs, 01 .. immed, 10 .. DataBus, 11 .. aluout
* **aluop**
  + izbira AL operacije :
    - 00 .. AND, 01 .. OR, 10 .. ADD, 11 .. SUB
* **datawrite**
  + vpis v ali branje iz pomnilnika
    - 0 .. branje, 1 .. pisanje
* **irload**
  + vpis v ukazni register (IR)
    - 1 .. vpis v IR, 0 .. IR pomni
* **addrsel**
  + izbira vira za naslovno vodilo
    - 00 .. PC, 01 .. immed, 10 .. Rs, 11 .. Rd
* **pcload**
  + vpis v PC register
    - 1 .. vpis v PC, 0 .. PC pomni
* **pcsel**
  + izbira vhoda za vpis v PC register
    - 0 .. immed, 1 .. PC+1

# Naslovni prostor in V/I napravi

Naslov sestavlja 16 bitov in določa naslednji prostor vseh možnih naslovov :

|  |  |  |  |
| --- | --- | --- | --- |
| Naslov | | Naprava | Opis |
| Dec. | Hex. |
| 00000-16383 | 0000-3fff | RAM pomnilnik | 14 bitni naslov, pomeni 2^14 pomn. besed |
| 16384-32767 | 4000-7fff | LED zaslon 16x16 | 16 x 16 bitnih registrov – vsak bit predstavlja eno točko na zaslonu |
| 32768-49151 | 8000-bfff | Znakovni zaslon  TTY | 7 bitni register za vpis ASCII kode znaka za prikaz na zaslonu |
| 49152-65535 | c000-ffff |  |  |

Sistem uporablja t.i. pomnilniško preslikane V/I naprave. To pomeni, da so registri naprav vidni kot običajno pomnilniške besede, vendar na posebnih naslovih. Tako vedno dostopamo do natanko ene V/I naprave oziroma RAM pomnilnika. Ob dostopih v prvo skupino naslovov se aktivira in odziva RAM pomnilnik, ob vpisu na naslednji dve skupini naslovov pa se aktivirata izhodni prikazovalni napravi (LED zaslon in znakovni zaslon TTY). Primer izpisa na prikazovalni napravi predstavlja preizkusni program za V/I naprave.

# Preizkusni programi

Za pomoč pri razumevanju delovanja sistema in njegovem programiranju je v nadaljevanju nekaj primerov programov v zbirniku. Za prevedbo v strojne ukaze oziroma vsebino pomnilnika se lahko uporabi preprost zbirnik, realiziran v Excelu (MiniMiMo\_Assembler.xlsx). Programi se po prevajanju o zapisani tudi v datotekah s končnico .ram, ki se lahko prenesejo direktno v RAM pomnilnik modela in se potem izvajajo (pritisk na F9 pomeni izvedbo ene urine periode). Za vzpostavitev začetnega stanja je na voljo tipka Reset. Prenos vsebine datoteke v RAM pomnilnik prikazuje spodnja slika :

Slika, ki vsebuje besede besedilo

Opis je samodejno ustvarjenSlika, ki vsebuje besede miza

Opis je samodejno ustvarjen

## Preizkusni program: Seštej tabelo

* V pomnilniku od naslova 0x20 naprej se nahajajo 16-bitna števila. Zadnje število ima vrednost 0.
* Program naj sešteje vsa števila in vsoto zapiše na naslov 0x40 in potem počaka v mrtvi zanki
* Uporabimo 4 registre:
  + R0 kaže na seznam števil (0x20)
  + R1 vsebuje tekočo vsoto
  + R2 vsebuje prebrano število
  + R3 vsebuje 1 za povečevanje R0

|  |  |  |  |
| --- | --- | --- | --- |
| Naslov | Oznaka | Ukaz v zbirniku | Strojni ukaz |
| 0x0000 |  | MOV  R1, #0x00 | e400 |
| 0x0001 |  | MOV  R0, #0x20 | e020 |
| 0x0002 |  | MOV R3, #0x01 | ec01 |
| 0x0003 | loop: | LDRH  R2, [R0] | 4800 |
| 0x0004 |  | BEQ R2, end | 8808 |
| 0x0005 |  | ADD R1, R1, R2 | 2600 |
| 0x0006 |  | ADD R0, R0, R3 | 2300 |
| 0x0007 |  | B loop | ff03 |
| 0x0008 | end: | STRH  R1, [#0x40] | d440 |
| 0x0009 | Inf: | B inf | ff09 |

Program se nahaja v datoteki z vsebino RAM pomnilnika - minimimo\_sestej.ram.

## Preizkusni program: V/I naprave

* V pomnilniku se od naslova 0x4000 naprej nahaja znakovni terminal TTY, od naslova 0x8000 naprej pa 16x16 LED matrika
* Program naj vpisuje v obe V/I napravi vrednosti, ki se ves čas povečujejo
* sešteje vsa števila in vsoto zapiše na naslov 0x40 in potem počaka v mrtvi zanki
* Uporabimo 4 registre:
  + R0 vsebuje 1 za povečevanje
  + R1 vsebuje začetno ASCII kodo (64 = '@')
  + R2 vsebuje začetni naslov 16. vrstice matrike LED (16399)
  + R3 vsebuje začetni naslov znakovnega terminala TTY (32768)

|  |  |  |  |
| --- | --- | --- | --- |
| Naslov | Oznaka | Ukaz v zbirniku | Strojni ukaz |
| 0x0000 | main: | MOV  R0, #1 | e001 |
| 0x0001 |  | MOV  R1, #64 | e440 |
| 0x0002 |  | MOV  R2, #0x20 | e820 |
| 0x0003 |  | LDRH  R2, [R2] | 4a00 |
| 0x0004 |  | MOV  R3, #0x21 | ec21 |
| 0x0005 |  | LDRH  R3, [R3] | 4f00 |
| 0x0006 | loop: | STRH  R1, [R2] | 5600 |
| 0x0007 |  | STRH  R1, [R3] | 5700 |
| 0x0008 |  | ADD R1, R1, R0 | 2400 |
| 0x0009 |  | B loop | ff06 |
| … |  |  |  |
| 0x0020 |  | 0x8000 = 32768 | TTY |
| 0x0021 |  | 0x400f = 16399 | LED |
|  |  |  |  |

Program se nahaja v datoteki z vsebino RAM pomnilnika - minimimo\_testIO.ram.

## Preizkusni program: vsota dveh števil

* V pomnilniku se na naslovih 32 (0x20) in 33 (ox21) nahajata dve 16-bitni števili
* Program naj sešteje števili in vsoto zapiše na naslov 34 (0x22)
* Po koncu naj potem program počaka v mrtvi zanki
* Uporabimo 3 registre:
  + R0 je bazni register
  + R1 vsebuje prvo število
  + R2 vsebuje drugo število

|  |  |  |  |
| --- | --- | --- | --- |
| Naslov | Oznaka | Ukaz v zbirniku | Strojni ukaz |
| 0x0000 | main: | MOV  R0, #0x20 | e020 |
| 0x0001 |  | LDRH  R1, [R0] | 4400 |
| 0x0002 |  | MOV  R0, #0x21 | e021 |
| 0x0003 |  | LDRH  R2, [R0] | 4800 |
| 0x0004 |  | ADD R2, R2, R1 | 2900 |
| 0x0005 |  | MOV  R0, #0x22 | e022 |
| 0x0006 |  | STRH  R2, [R0] | 5800 |
| 0x0007 | inf: | B inf | f007 |
| … |  |  |  |
| 0x0020 |  | 0x10 = 16 | 0010 |
| 0x0021 |  | 0x40 = 64 | 0040 |
| 0x0022 |  | ? = 0x50 = 80 | 0050 |

Program se nahaja v datoteki z vsebino RAM pomnilnika - minimimo\_vsota.ram.