

# 8-bitni računalnik

Nikola Sekulovski

# Kazalo

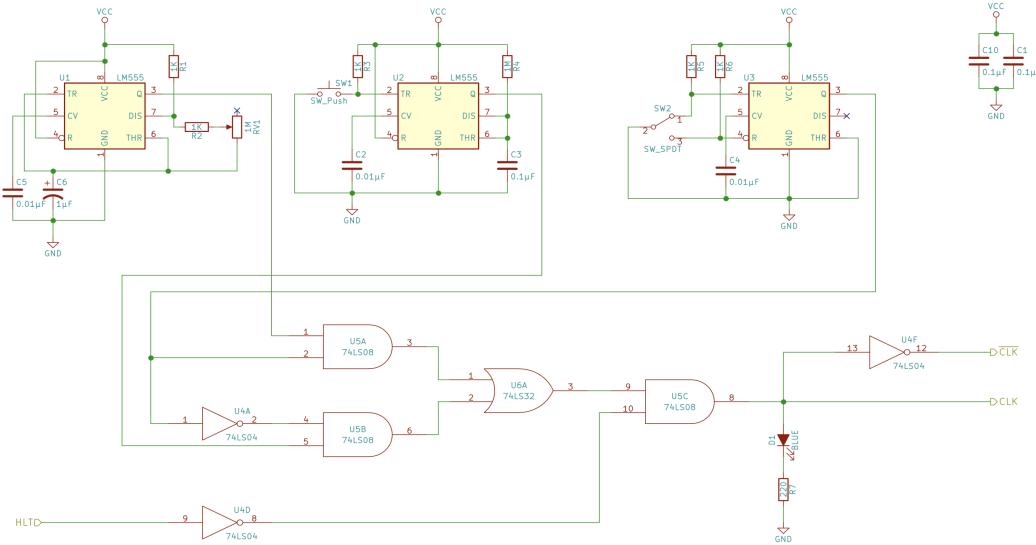
<b>1 Celoten seznam delov</b>	<b>3</b>
<b>2 Modul za uro</b>	<b>4</b>
<b>3 Registri</b>	<b>5</b>
3.1 Register A in B . . . . .	5
3.2 Register ukazov . . . . .	5
<b>4 Aritmetična logična enota (ALU)</b>	<b>7</b>
<b>5 RAM</b>	<b>8</b>
<b>6 Programski števec</b>	<b>10</b>
<b>7 Izhodni register</b>	<b>11</b>
<b>8 Povezovanje vsega na vodilu</b>	<b>12</b>
<b>9 Kontrolna logika</b>	<b>13</b>
<b>10 Primer programa</b>	<b>15</b>

# 1 Celoten seznam delov

Število kosov	Opis
12	Breadboard
10	1kΩ
9	10kΩ
1	100kΩ
24	470Ω
1	1MΩ
1	1MΩ potentiometer
6	0.01µF
16	0.1µF
1	1µF
4	555 timer IC
2	74LS00 (Quad NAND gate)
1	74LS02 (Quad NOR gate)
5	74LS04 (Hex inverter)
3	74LS08 (Quad AND gate)
1	74LS32 (Quad OR gate)
1	74LS107 (Dual JK flip-flop)
2	74LS86 (Quad XOR gate)
1	74LS138 (3-to-8 line decoder)
1	74LS139 (Dual 2-line to 4-line decoder)
4	74LS157 (Quad 2-to-1 line data selector)
2	74LS161 (4-bit synchronous binary counter)
8	74LS173 (4-bit D-type register)
2	74189 (64-bit random access memory)
6	74LS245 (Octal bus transceiver)
1	74LS273 (Octal D flip-flop)
2	74LS283 (4-bit binary full adder)
3	28C16 EEPROM
3	2-položajno stikalo
3	gumb
1	8-položajno stikalo
1	4-položajno stikalo
44	rdeče svetleče diode
8	rumene svetleče diode
12	zelene svetleče diode
21	modre svetleče diode
4	7-segmentni zaslon

## 2 Modul za uro

Ura (slika 1) računalnika se uporablja za sinhronizacijo vseh operacij. Ura, ki jo sestavljamo, temelji na priljubljenem časovniku 555 IC. Naša ura ima nastavljivo hitrost (od manj kot 1 Hz do nekaj sto Hz). Uro lahko preklopite tudi v ročni način, v katerem s pritiskom na gumb pospešite vsak cikel ure.



Slika 1: Ura

Časovnik na levi strani je astabilen. To je glavna ura računalnika. Časovnik na sredini se uporablja kot debounce filter, da bi lahko izvajali ročne impulze ure. Časovnik na desni strani je bistabilen. Zaradi komparatorja v notranjosti se uporablja kot debounce filter za stikalo, ki se uporablja za prehod iz samodejnega v ročni način za uro.

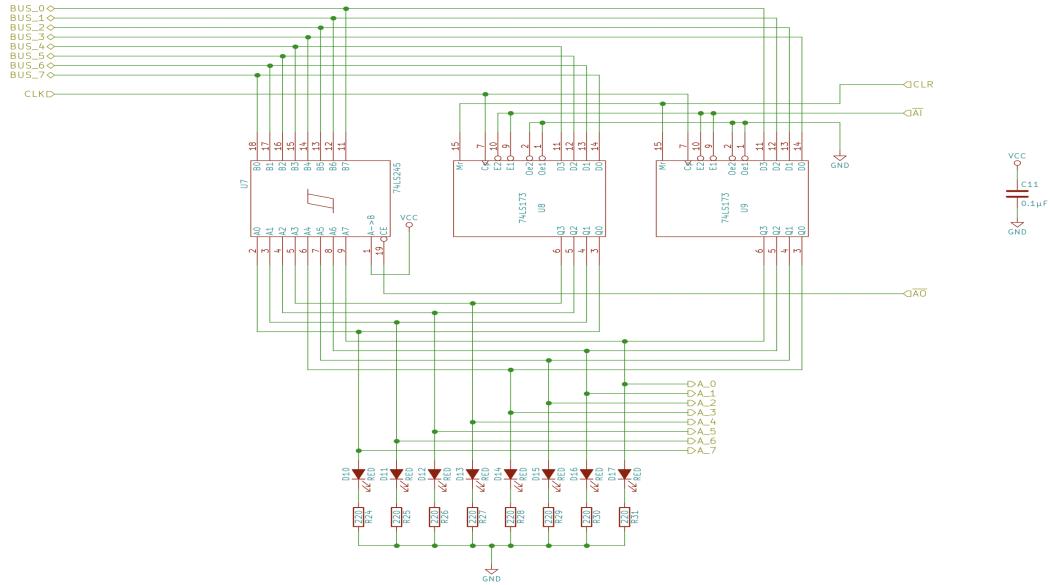
### 3 Registri

Večina procesorjev ima več registrov, v katerih so shranjene majhne količine podatkov, ki jih procesor obdeluje. V našem preprostem procesorju na ploščici bomo zgradili tri 8-bitne registre: A, B (slika 2) in IR (slika 3). Registra A in B sta registra za splošno uporabo. IR (register navodil) deluje podobno, vendar ga bomo uporabljali le za shranjevanje trenutnega navodila, ki se izvaja.

Registra A in B sta popolnoma enaka<sup>1</sup>, medtem ko je register navodil nekoliko drugačen. Edina razlika je, da so prek transceiverja na vodilo priključeni le širje najmanj pomembni biti, drugi širje biti pa bodo povezani z EEPROMe. Širje najpomembnejši biti registra navodil predstavljajo navodila, širje najmanj pomembni biti pa predstavljajo naslov, kot bo razvidno pozneje.

Čeprav imajo registri tipa D (74LS173) pine CE (chip enable) za povezavo z vodilom, jih ne bomo uporabljali. Ti pini bodo aktivni, da bodo registri stalno oddajali svoje podatke, namesto tega pa bomo dodali sprejemnik vodila (74LS245). Razlog za to je, da lahko na izhode registrov priključimo svetleče diode in tako vedno vidimo, kaj shranjujejo, da bi olajšali odpravljanje težav.

#### 3.1 Register A in B

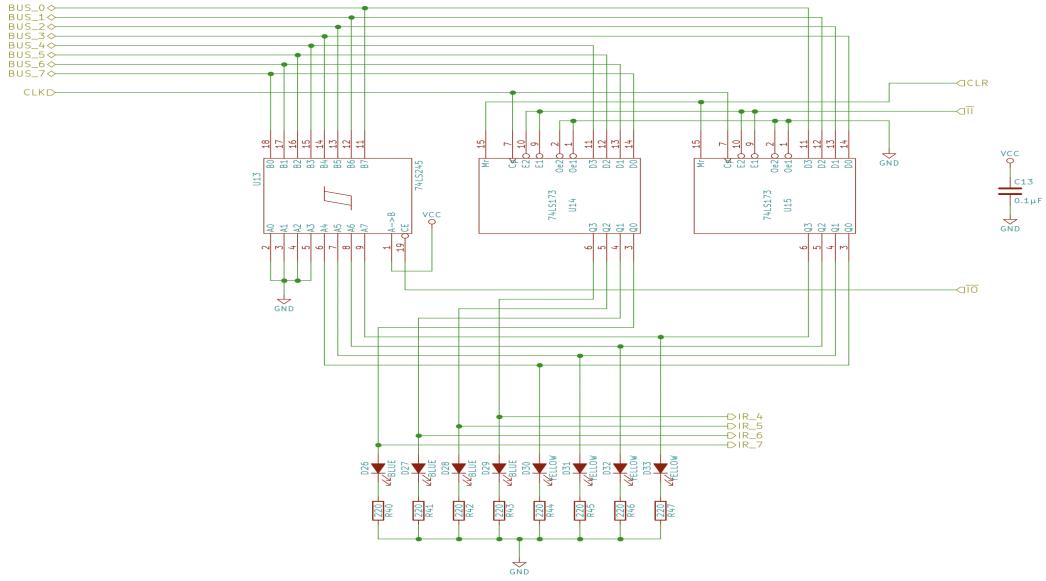


Slika 2: Register A in B

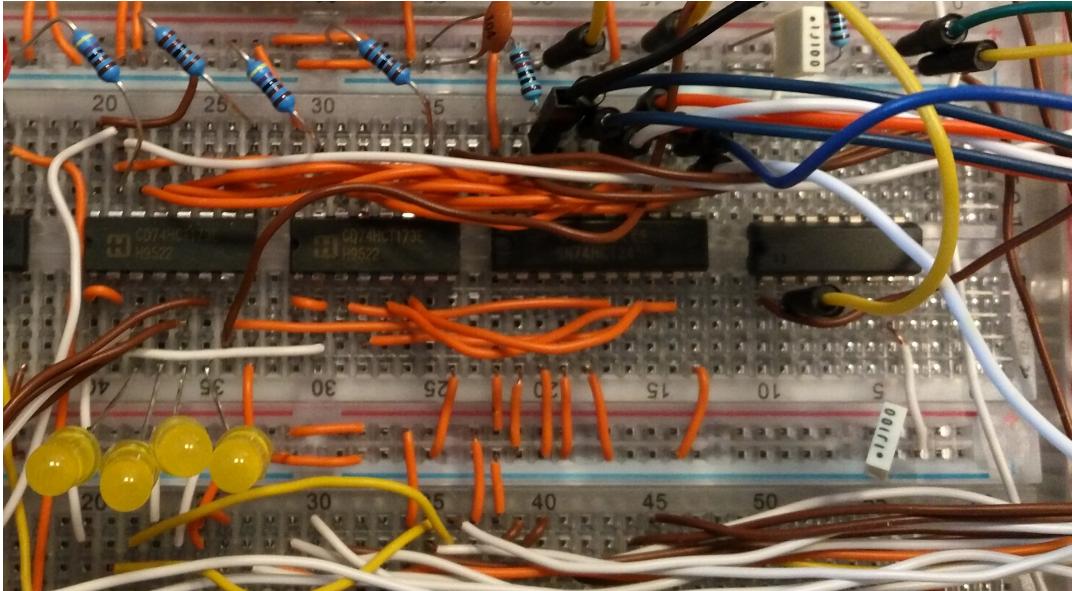
#### 3.2 Register ukazov

---

<sup>1</sup>74LS173 ni nič drugega kot flip flop, ki ima CE pin za nadzor izhoda. Ker lahko vsak čip shrani le 4 bite, bomo uporabili dva. Ura in vhodna nožica obeh čipov sta seveda povezani skupaj.



Slika 3: Register ukazov

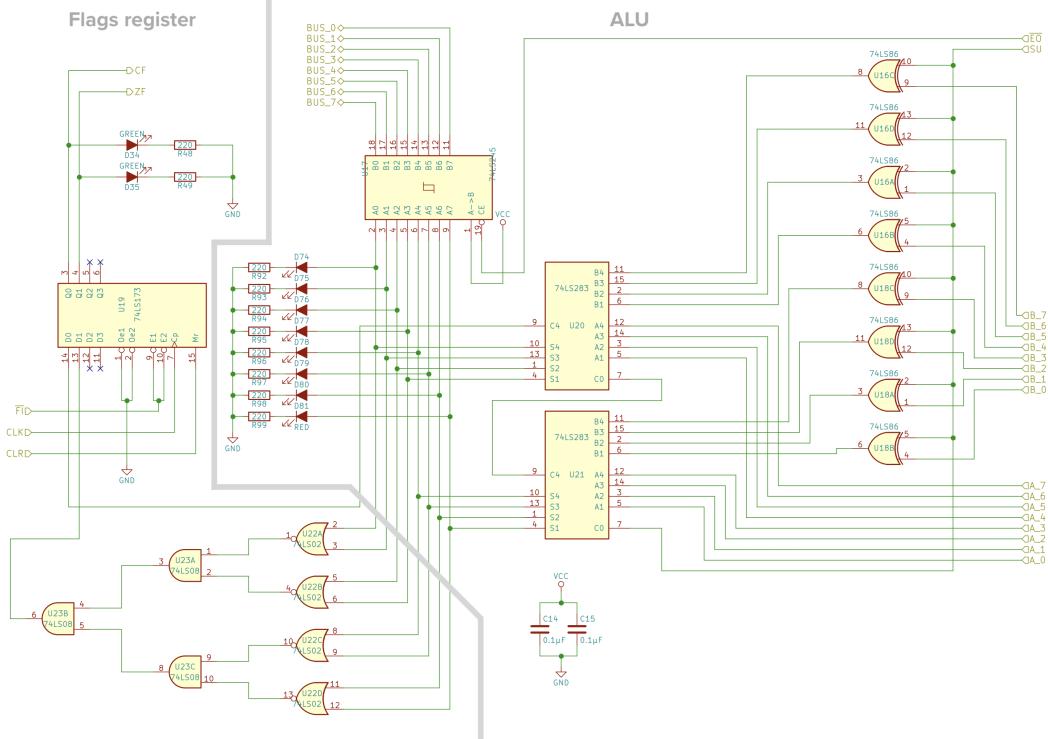


Slika 4: Register ukazov zgrajen

## 4 Aritmetična logična enota (ALU)

Aritmetično-logična enota (slika 5) v procesorju je običajno sposobna izvajati različne aritmetične, bitne in primerjalne operacije z binarnimi števili. V našem preprostem procesorju na plošči ALU lahko samo sešteva in odšteva. Povezana je z registrom A in B ter izpisuje bodisi vsoto A+B bodisi razliko A-B.

Ker uporabljamo binarne seštevalnike (74LS283), je funkcija seštevanja precej preprosta za razumevanje. Da bi lahko odšteli dve števili, se vrednosti iz registra B pretvorijo v dvojiški komplement. Eniški komplement lahko preprosto dosežemo z uporabo vrat XOR (74LS86). Če izhode iz registra B najprej priključimo na vrata XOR, katerih drugi vhod je logična 1, dobimo njihove invertirane vrednosti. To je funkcija signala SU. Po drugi strani pa, kadar je vrednost 0, gredo vsi vhodi, ki prihajajo iz registra B (izhodi registra B), neposredno v seštevalnike. Dvojiški komplement dosežemo tako, da signal SU povežemo s prenosom v drugi seštevalnik (seštevalnik za štiri najmanj pomembne bite). Tako se invertiranim vhodom doda 1, s čimer dobimo njihov dvojiški komplement.



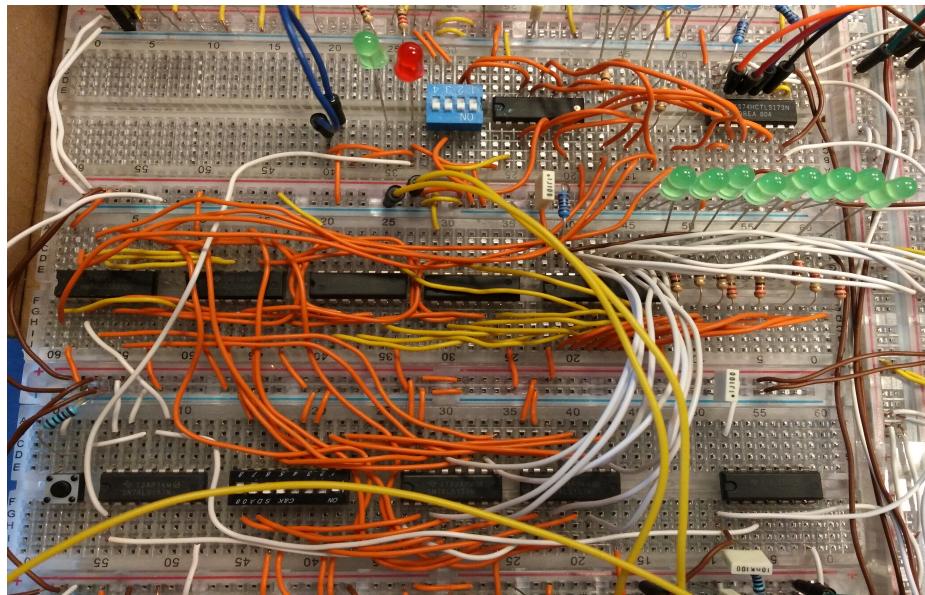
Slika 5: Aritmetična logična enota

## 5 RAM

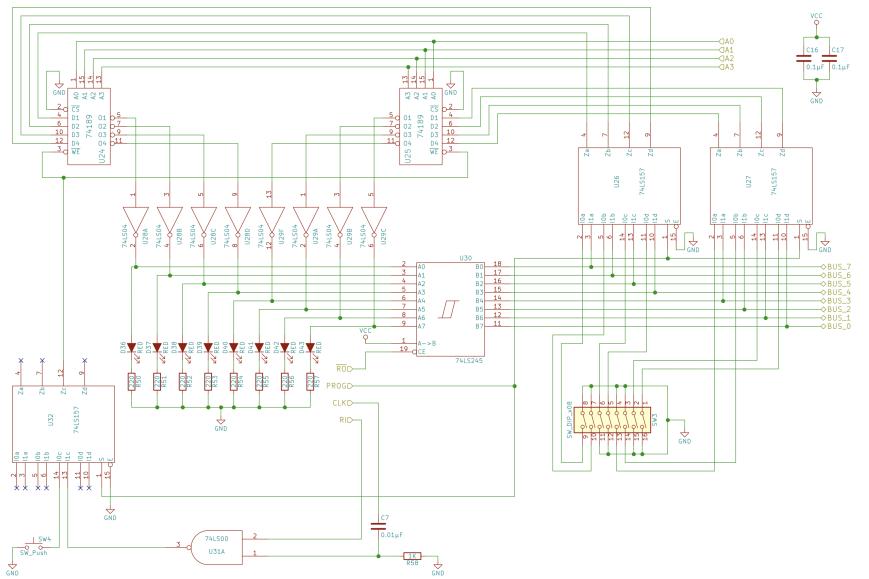
V pomnilnik z naključnim dostopom (RAM) je shranjen program, ki ga izvaja računalnik, in vsi podatki, ki jih program potrebuje. Naš tablični računalnik uporablja 4-bitne naslove, kar pomeni, da ima na voljo le 16 bajtov pomnilnika RAM, kar omejuje velikost in zapletenost programov, ki jih lahko izvaja. To je daleč največja omejitev računalnika.

Register naslovov pomnilnika (slika ??) se uporablja, da lahko ročno izberete naslov pomnilnika RAM s 4-položajnim stikalom ali da samodejno naložite vrednosti naslova z vodila v register (74LS173), kjer je začasno shranjen. Preklop med ročno in samodejno funkcionalnostjo se izvede s preprostim stikalom.

Modul ram lahko sprejema podatke z vodila ali pa jih vanj ročno vnese uporabnik s pomočjo 8-položajnega stikala. V ročnem načinu lahko uporabnik ročno izbere naslov in vrednost, ki bo zapisana. To je pomembno, saj služi kot sredstvo za programiranje računalnika. Podatki se v pomnilnik RAM zapišejo s pritiskom na gumb. Za ročno zapisovanje podatkov uporabljammo vrata NAND, ki preverjajo, ali je stikalo iz pomnilniškega naslovnega registra v ročnem načinu, in zagotavljajo, da se podatki zapišejo ob naraščajočem robu. Kot detektor naraščajočega roba smo uporabili preprosto RC vezje.



Slika 6: Izgradnja rama

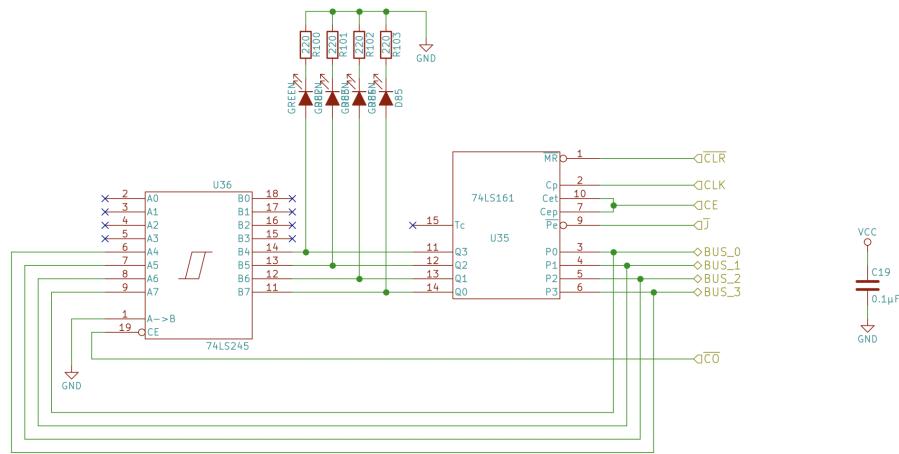


Slika 7: RAM

## 6 Programski števec

Programski števec (PC) šteje v binarni obliki in tako spremlja, katero navodilo računalnik trenutno izvaja.

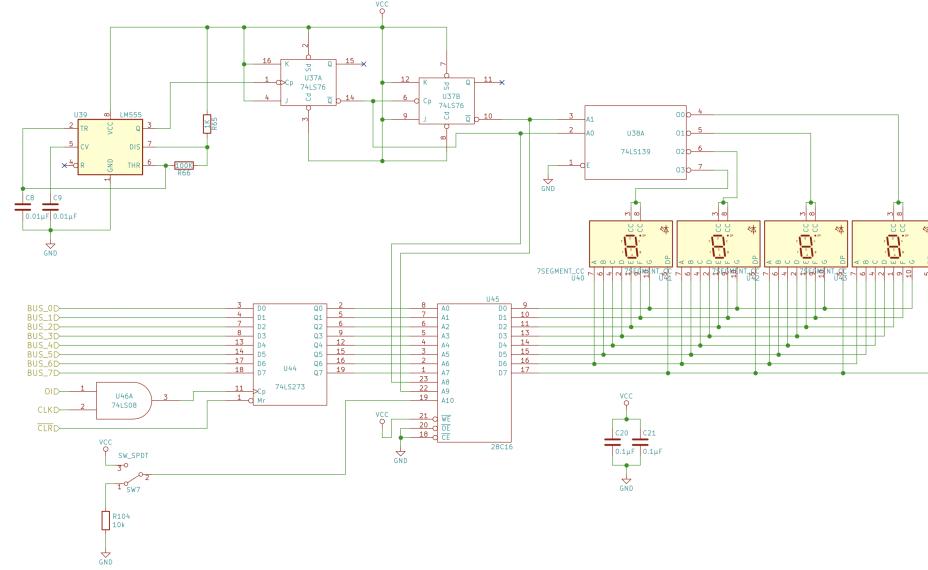
Funkcija programskega števca deluje enako kot drug register. Šteje od 0 do 15, njegova binarna vrednost pa predstavlja naslednje navodilo našega računalnika. Števec lahko izvaja tudi skoke, tj. preide na določene vrednosti, ki jih prevzame z vodila. To je pomembno, saj bi nam omogočilo zanke v našem programu.



Slika 8: Programski števec

## 7 Izhodni register

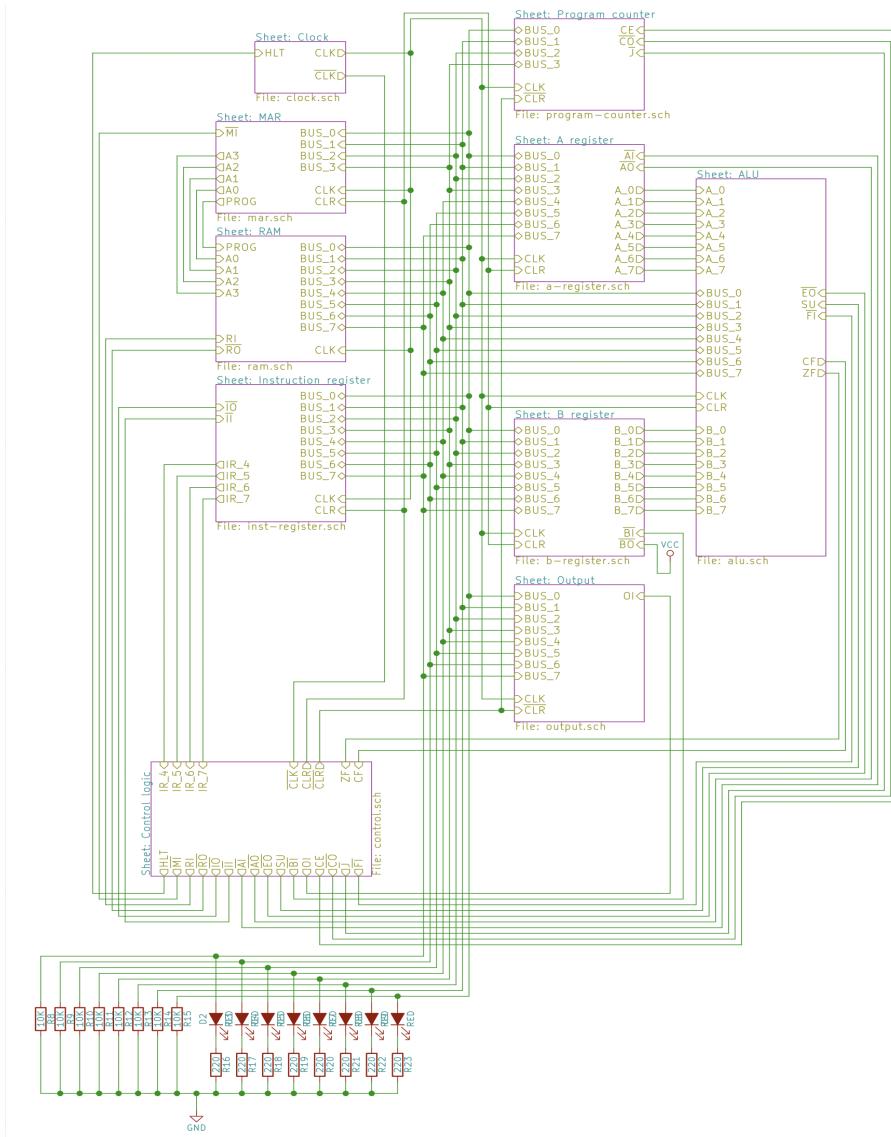
Izhodni register (slika 9) je podoben kateremu koli drugemu registru (kot sta registra A in B), le da se njegova vsebina ne prikazuje v binarni obliki na 8 svetlečih diodah, temveč v desetiški obliki na 7-segmentnem zaslonu.



Slika 9: Izhodni register

## 8 Povezovanje vsega na vodilu

Pred izgradnjo nadzorne logike želimo vse module povezati s skupnim vodilom in preizkusiti delovanje (slika 10). Modularnost zasnove olajša testiranje vsakega modula posebej, tako da ne bomo nikoli prišli do točke, ko bomo vse skupaj sestavili in nič ne bo delovalo.



Slika 10: Pogled na višji nivo

## 9 Kontrolna logika

Krmilna logika (slika 11) je srce procesorja. Določa opkode, ki jih procesor prepozna, in kaj se zgodi, ko izvede posamezno navodilo.

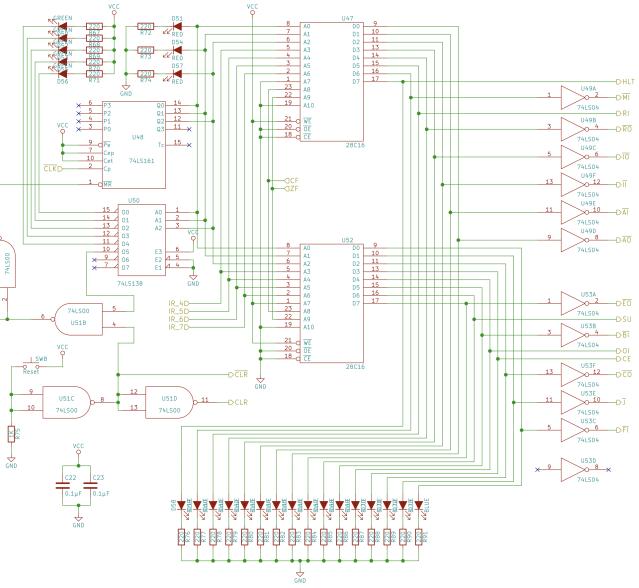
Za nadzorno logiko računalnika uporabljamo dve EEPROM, ki služita kot preglednici za naslednja navodila. Vsako navodilo je sestavljeno iz mikroinstrukcij. Pri moji zasnovi ima lahko eno navodilo največ 5 mikroinstrukcij. Če programski števec spremišča navodila, potrebujemo še en števec, ki spremišča mikroinstrukcije. V ta namen imamo še en čip 74LS161, ki je priključen tako, da se ponastavi na vsakih 5 števcev. Tri izhodne vrednosti iz števca gredo v dekoder. Obe EEPROM-ki sta programirani z enakimi podatki, vendar je naslovni nožič A7 na eni EEPROM-ki priključen na 0 V, na drugi pa na 5 V. Tako lahko vsaka EEPROM poskrbi za osem različnih krmilnih signalov, saj imata osem izhodov. Pini A6 do A3 so povezani s štirimi najpomembnejšimi biti registra navodil. Vtiči A2 do A0 so povezani z izhodom mikro števca navodil. Ostali naslovni nožice so priključeni na 0 V. Izhodi EEPROM so povezani s 16 svetlečami diodami, da lahko uporabnik vizualno vidi, katero je naslednje navodilo. Od tam gredo nekateri signali na inverterje, nekateri pa ne. Razlog za to je, da nekateri čipi za aktivacijo potrebujejo invertiran signal, nekateri pa ne. Na primer, da bi v register A vnesli podatke, mora biti njegov vhodni signal nizek, po drugi strani pa mora biti za omogočanje štetja programskega števca signal za omogočanje visok. Na podlagi preglednice iskanja v EEPROM lahko uporabnik programira računalnik tako, da ročno postavi vrednosti na različne naslove v pomnilniku RAM. Prvi štirje biti predstavljajo ukaz, drugi štirje pa naslov.

Za večjo praktičnost je bil dodan gumb za ponastavitev, ki ponastavi vsak register, števce in ALU. Ker nekateri čipi za ponastavitev potrebujejo nizek signal, nekateri pa visok, se v ta namen uporablja vrata NAND. Namesto inverterja je uporabljeno NAND-vratce, saj je bilo na 74LS00 nekaj neizkoriščenih pinov.

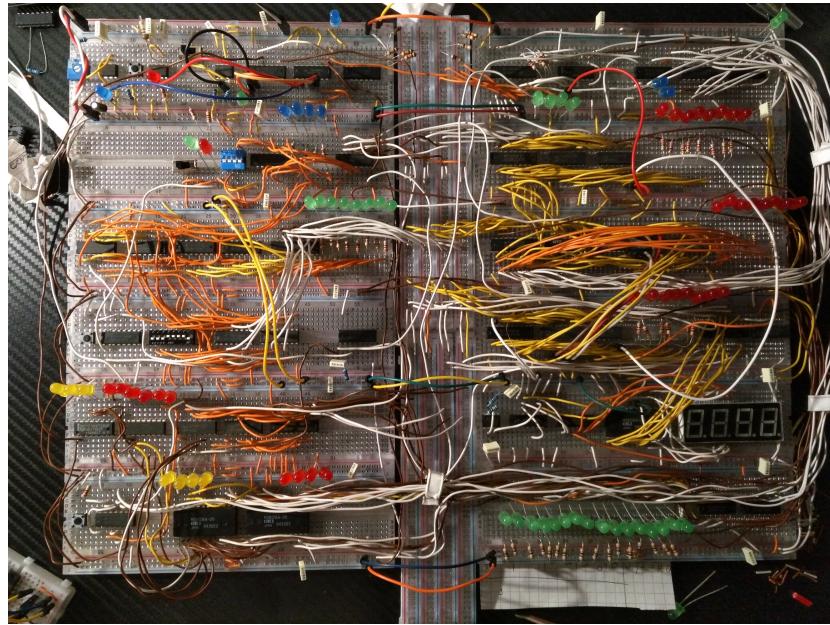
Da bi bil računalnik Turingov popoln<sup>2</sup>, mora biti sposoben izvajati pogojne stavke. V ta namen dodamo register zastavic. Njegov namen je prepozнатi, kdaj imamo v ALU 0 ali kdaj presežemo omejitev 255 vrednosti ALU. Ko je izpolnjen eden ali oba od teh pogojev, odvisno od programa, lahko računalnik izvede drugačno operacijo. Pri tem je pomembno omeniti, da sta dva signala iz registra povezana z dvema naslovima linijama pomnilnikov EEPROM. To bi računalniku omogočilo, da prepozna, kdaj je zastavica nastavljena, in preide na drugo vrstico v preglednici za iskanje.

---

<sup>2</sup>Seveda mora imeti neskončno količino pomnilnika, da bi bil Turingov računalnik popoln, vendar menim, da smo dovolj blizu. Pomnilnik se lahko po potrebi kasneje nadgradi.



Slika 11: Kontrolna logika



Slika 12: Končna verzija

## 10 Primer programa

- LDA 14 - 0001 1110 (naloži vsebino naslova 14 v register A)
  - CO (counter out) MI (memory in)
  - RO (ram out) II (instruction in) CE (counter enable)
  - IO (instruction out) MI (memory in)
  - RO (ram out) AI (a-register in)
- ADD 15 - 0010 1111 (doda vsebino iz naslova 15 vrednosti v registru A)
  - CO MI
  - RO II CE
  - IO MI
  - RO BI (b-register in)
  - EO (sum out) AI
- OUT - 1110 0000 (izpiše vrednost na 7-segmentni zaslon)
  - CO MI
  - RO II CE
  - AO (a-register out) OI (output register in)
- HLT - 1111 0000 (zavrti uro in program se ustavi)
- na naslovu 14 imamo številko 0001 1100
- na naslovu 15 imamo številko 0000 1110