Emulator Mikroprocesorja Motorola M680X

(Inovacijski predlog)

KAZALO VSEBINE

[1 Povzetek v](#_Toc150603295)

[2 Uvod 5](#_Toc150603296)

[3 RAZVOJ 1](#_Toc150603297)

[3.1 Izbira okolja za razvoj 1](#_Toc150603298)

[3.2 Testiranje zmogljivosti Qt okolja 2](#_Toc150603299)

[4 Model 3](#_Toc150603300)

[4.1 Uporabniški vmesnik 3](#_Toc150603301)

[4.1.1 Števec vrstic 4](#_Toc150603302)

[4.1.2 Polje za vpis assembly kode 5](#_Toc150603303)

[4.1.3 Polje za prikaz spomina 6](#_Toc150603304)

[4.1.4 Registri 7](#_Toc150603305)

[4.1.5 Stran z večmi zavihki 7](#_Toc150603306)

[5 ZAKLJUČEK 11](#_Toc150603307)

Napiši strukturo raziskovalne naloge. Poglavja in podpoglavja in podpod poglaje (če je treba)

1. Uvod
2. Razvoj - ???
   1. Moja zgodovina ravjnih okolij in uporabe porogramskih jezikov
   2. Izbira okolja za mojo nalogo
   3. Opis okolja qt
      1. Framework
         1. Kaj je famework
         2. Kaj je ide
      2. funkcije
      3. IDE
3. Razvoj
   1. Izhodišča
   2. Elementi emulatojra
   3. …
4. Trajnostni x, y, z
5. Zaključek

# Povzetek

Moj inovacijski predlog predstavlja razvoj emulatorja za mikro procesorja Motorola M6800 in Motorola M6803, ki omogoča uporabnikom, še posebej tistim, ki se želijo naučiti osnove nizko nivojskega programiranja in dela z »assembly« programskimi jeziki, edinstveno priložnost za izboljšanje njihovega razumevanja računalniške arhitekture. Emulator vsebuje uporabniku prijazen vmesnik s poljem za vnos »assembly« kode, prikaz spomina, orodja za sestavljanje in razstavljanje ukazov, orodja za izvajanje ukazov, zaslon, vhodne medpomnilnike, tabelo ukazov in njihove lastnosti, ter veliko nastavitev za olajšanje uporabe emulatorja.

# Uvod

Ta inovacijski predlog opisuje razvoj mojega emulatorja za mikroprocesor MC6800, ki ga je razvila in izdelala ameriška družba Motorola v 1974. Ta mikroprocesor je v družini mikroprocesorjev, ki se je imenoval »M6800 Microcomputer System« in so ga kasneje preimenovali v »68XX«. Ta procesor podpira 72 ukazov, upravljanje z 8-bitnimi ukazi ter 16 bitnimi spominskimi naslovi. (Motorola 6800 - Wikipedia, 2023).

Ta mikroprocesor ima velik potencial za učenje delovanja in upravljanja enostavnih računalnikov na nižjem nivoju programiranja. MC6800 ima samo 2 podatkovna registra(akumulatorja), en 16-bitni indeksni register, 16-bitni programski števec ter 16-bitni kazalec sklada. Torej ima vse osnovne komponente sposobnega mikroprocesorja, hkrati pa ima enostaven nabor ukazov in tako omogoča hitro učenje ter obvladanje mikroprocesorja.

Poznanje katerega koli programskega jezika assembly je ključno za globlje razumevanje delovanja računalnika na nižjem nivoju. Tudi pri programiranju na višjih programskih jezikih, razumevanje nižjih omogoči da razumete kako se ukazi »pod pokrovom« prevajajo in izvršujejo. To pa pomaga pri optimizaciji programov, odpravljanju varnostnih ranljivosti ter razvoju sistemov kot so gonilniki ali vgrajeni sistemi.

Odločil sem se ustvariti emulator MC6800, ki bo uporabo in učenje delovanja tega mikroprocesorja omogočil komurkoli. Menim; da je tak procesor primeren za učenje assembly jezika študentom, dijakom ali začetnikom programiranja.

# Načrt in priprave

## Osebna zgodovina progrmairanja.

//Pred začetkom tega projekta sem dobro poznal programske jezike c# in python. C# sem se naučil s programskim orodjem Unity. Tam sem naredil nekaj prototipov videoiger. S pythonom sem pa delal nekaj orodij za avtomatizacijo vsakdanjih opravil. Imel sem izkušnje tudi z mikroprocesorjema Arduino in Raspberry Pi.

## Obstoječi emulatorji.

### Motorola 6800 Simulator različica 1.33p 2.cR

Z MC8600 sem se prvič ukvarjal med poukom računalništva v srednji šoli. Uporabljali smo zastarel emulator procesorja Motorola MC6800 imenovan »Motorola 6800 Simulator različica 1.33p 2.cR«, katerega avtor je Darko Kropf. Ta emulator ima mnoge pomanjkljivosti. Ene izmed teh so:

* Urejevalnik deluje počasi in nepredvidljivo.
* Največja hitrost izvajanja, ki sem jo izmeril je bila okoli sedem ukazov na sekundo.
* Ima tudi omejen pomnilnik z le štiri tisoč šestindevetdesetimi naslovi.
* Ta emulator je povprečnemu uporabniku nadležen zaradi nepotrebnih omejitev sintakse (omejitev nazivov na 6 črk, nepotrebna pravila, kot je obezno definiranje imena programa s besedo »NAM«, prepoved ukazov za skoke, ki skačejo izven prevedenega programa, ipd.).
* Ta emulator je ustvarjen za operativni sistem MS-DOS.
* Pomnilnik je nepregleden.

Na naslednji sliki je prikazan v emulatorju MS-DOSa, ki se imenuje DOSBox.

A computer screen shot of a computer

Description automatically generated

### SDK6800/6811 Emulator v1.14 ([www.HVRSoftware.com](http://www.HVRSoftware.com))

Ta emulator sem našel sam, je velik napredek od prej omenjenega saj tudi naravno deluje na OS Windows 10. Ukazi se izvršujejo veliko hitreje. Ima tudi možnost da ne prikazuje vseh informacij o trenutnem izvajanju, ki pospeši izvrševanje. Ko prikazuje vse informacije izvaja približno 256 ukazov na sekundo. Ko pa ne prikazuje informacij(razen zaslona in polja za prikaz spomina) pa približno 1500 ukazov na sekundo.

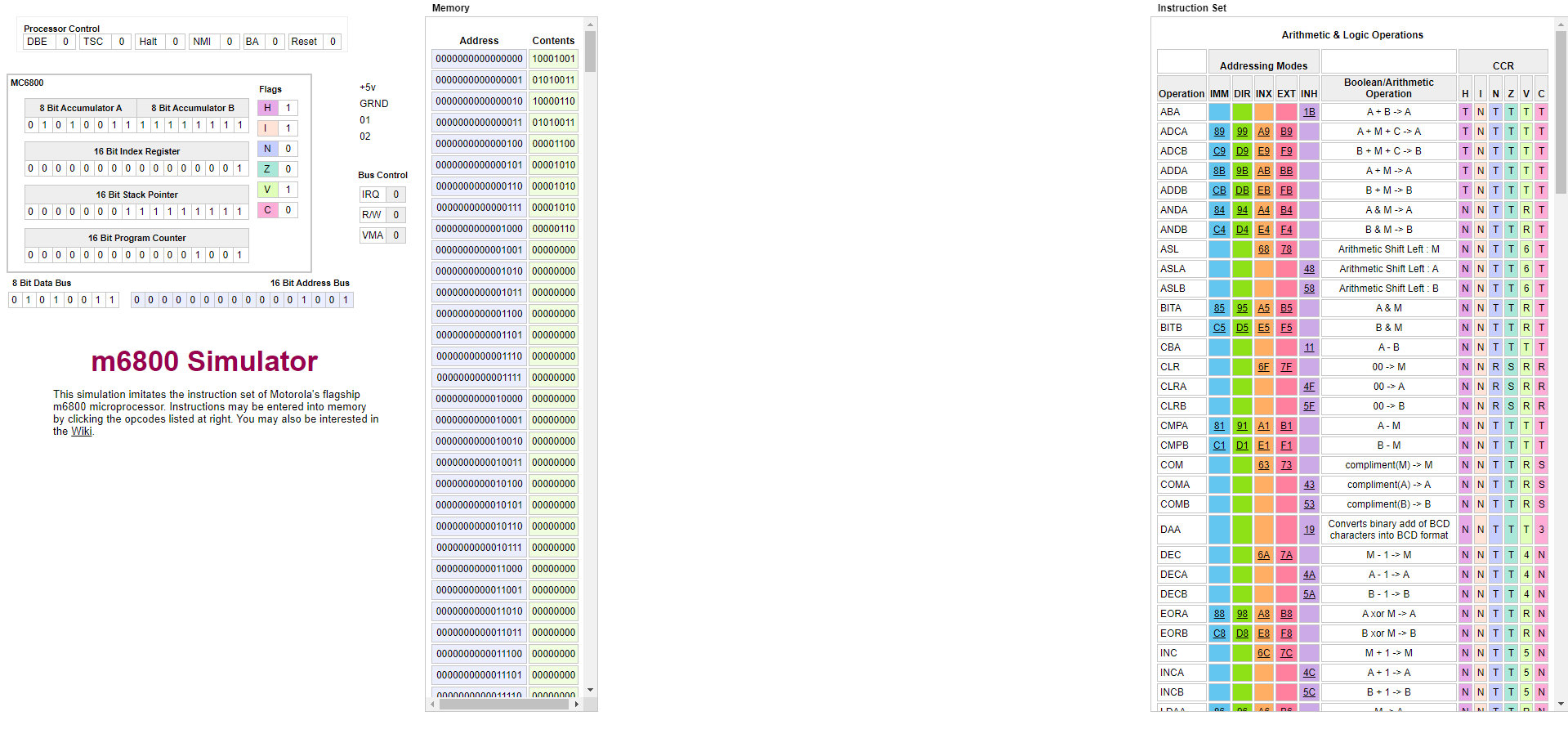
* Ta emulator je tudi dokaj nepredvidljiv, saj pod neznanimi pogoji samodejno izvajanje ne deluje.
* Na koncu samodejnega izvajanja se tudi emulator samodejno ponastavi kar onemogoči pregled spremenljivk ali registrov po izvajanju nekega programa.
* Ta emulator tudi ne podpira izvrševanje po ciklih.
* Nekateri ukazi ne delujejo pravilno.

A screenshot of a computer

Description automatically generated

### Ostale alternative

Našel sem še nekaj emulatorov, ki jih nisem uspel niti zagnat. Našel sem pa enega ki deluje preko spletne strani. Ta nima assemblerja saj deluje tako, da uporabnik s klikom na operacijsko kodo vstavi ukaz v spomin in ga izvrši.



## Načrt

Projekt sem začel z grobo skico izgleda emulatorja. Odločil sem se, da bo uporabniški vmesnik podoben prej omenjenem emulatorju »SDK6800/6811 Emulator«. Postavil sem si tudi cilje in kriterije kateri bi zagotovili, da je moj emulator boljši in uporabnejši od že obstoječih. Prvotni cilj projekta je razvoj popolnoma funkcionalnega emulatorja za procesor Motorola M6800. V njega bom vključil vse ali vsaj večino funkcij ter orodij, ki so že na voljo ter jih dopolnil in dodal svoje.

Za dosego teh ciljev sem določil ključne kriterije za uspeh. Prvič, emulator mora zagotavljati natančno emulacijo, ki omogoča pravilno izvajanje M6800 instrukcij. Hitrost emulacije je prav tako ključna, saj želim, da je emulator dovolj hiter za tekoče izvajanje programov. Stabilnost delovanja je pomembna za preprečevanje nepredvidenih napak, medtem ko je oblikovanje uporabniškega vmesnika ključno za enostavno upravljanje z emulatorjem.

## Iskanje primernega programskega okolja

Večina programov se razvija v obstoječih programskih okoljih, kot so WinForms, Qt, JavaFX, WPF, Xamarin, Swift in podobni, zaradi lažje in hitrejše uporabe. Čeprav nekateri programerji nizkih jezikov ustvarjajo lastna programska okolja za svoje specifične uporabe, je to običajno mnogo zahtevnejše delo.

Zaradi mojih izkušenj z C jeziki sem najprej iskal okolje ustvarjeno za C#

# RAZVOJ

Emulator je bil razvit z ogrodjem za programsko opermo »Qt«, ki je odprtokodno in zmogljivo okolje za razvoj grafičnih uporabniških vmesnikov ter ostale programske opereme. Večina emulatorja je bila izdelana z programskim jezikom c++. Nekaj postopkov razvoja sem avtomatiziral z skriptami napisanimi v programskem jeziku python.

## Izbira okolja

Prva različica emulatorja je bila izdelana v programskem okolju WinForms s programskim jezikom c#. V njej sem deloma dokončal prevajalnik ukazov. Izvrševanje ukazov je podpiralo le nekaj osnovnih ukazov kot so »LDA« in tele samo pri vsebovanem naslavljanju.

A screenshot of a computer

Description automatically generated

Ta različica je bila zapuščena, saj je WinForms zastarelo ogrodje ter ne podpira hitrega spreminjanja elementov uporabniškega vmesnika. Naslednji poskus je bil v programskem okolju WPF(Windows Presentation Foundation), to različico sem žal izgublil, vendar tudi ta ni izpolnila zahtev po hitrosti izvajanja. Med iskanjem alternativ sem opazil možnost v Qt okolju.

## Testiranje zmogljivosti Qt okolja

Za preverjanje primernosti okolja Qt za moj projekt sem ustvaril osnovni model emulatorja, ki vključuje ključne komponente, kot so okno za vpis kode, prikaz pomnilnika ter dva akumulatorja. Postavil sem kriterije, med katerimi je ključna hitrost sprememb vrednosti akumulatorjev – želel sem, da emulator zmore spreminjati vrednost akomulatorja več kot 1000-krat v eni sekundi. Ta meritev je ključna za oceno učinkovitosti emulatorja v realnem času.

Hkrati sem postavil druge kriterije, vključno z odzivnostjo vmesnika, enostavnostjo prilagajanja uporabniškega vmesnika ter prilagodljivosti kode. Uspešno sem potrdil, da Qt presega vse postavljene kriterije, kar potrjuje njegovo primernost za nadaljnji razvoj emulatorja.

## Razvoj prve različice emulatorja.

Odločil sem se da bom emulator razvil modularno in to tako da najprej dokončam prevajanje ukazov in se nato lotim izvajanja ukazov in drugih stvari. Pravilnost prevajanja sem preverjal s pomočjo dokumenta »M6800 Programming reference manual«, ki je bil izdan leta 1976. Največ težav pri prevajanju sem imel z nazivi. Ukaz ali vrstica v sestavljanju z M6800 lahko ima naziv(label) drug ukaz lahko kliče ta naziv in dobi neko pomnilniško lokacijo ali vrednost. Težava se pojavi ko nek ukaz kliče naziv ki še ni bil definiran. Primer:

A screenshot of a computer

Description automatically generated

Ukaz »JMP« ali skok ima za operand »label«. Ko se ta koda prevaja bo v strojni kodi začasno bil operand prazen. Ko se pa enkrat vsi ukazi zapišejo in se njihovi nazivi shranijo, pa prevajalnik znova gre skozi vse ukaze ki so klicali nazive in v prazna mesta operandov zapiše ustrezno vrednost.

Na koncu sem v prevajalnik vključil direktive assemblerja, to so navodila, ki jih assemblerju podamo, da nadzorujejo proces sestavljanja. Moj emulator podpira naslednje:

1. .ORG(izhodišče): Določi izhodiščni ali začetni naslov programa ali odseka. To se zgodi tako da assembler sestavi sledeče ukaze na določenem pomnilniškem naslovu.
2. .RMB: Rezervira spomin oziroma preskoči pomnilniške lokacije med sestavljanjem.
3. .EQU: Nazivu pod katerim je klican .EQU je prirejena vrednost katero določi uporabnik. V primeru: »test .EQU 24« bo naziv test vseboval število 24.
4. .BYTE: V trenutnem naslovu sestavljanja se shrani en ali več zaporednih bajtov. Primer:

A white background with black text

Description automatically generated

1. .WORD: V trenutni naslov sestavljanja se shrani ena ali več besed(2 bajta).
2. .SETB: V podan naslov shrani podan bajt. Primer: ».SETB $FB00,24«.
3. .SETW: V podan naslov shrani besedo.
4. .STR: Shrani niz ali več nizov v trenutni naslov. ».STR "primer1","primer2"«

Dodal sem tudi konzolo ki kaže napake, opozorila in podobno ter polje, ki označuje vrstice v polju za vpis kode. Ustvaril sem tudi nekaj osnovnih nastavitev.

Nato sem se lotil programiranja delovanja procesorja M6800. Dodal sem vse registre in okenca ki jih prikazujejo ter gumbe, ki upravljajo s procesorjem. Dodal sem tudi gumbe, ki kodo shranjujejo in nalagajo.

Ko sem dodal še možnost direktnega urejanja spomina sem zraven dodal še sistem označevanja ukazov, ki označijo napake, naslednji ukaz, zabeležene ukaze ter lokacijo urejanja spomina.

Po veliko optimizacije in prilagajanja uporabniškega vmesnika sem dodal tudi ukaze ter assembler za procesor M6803. Začel sem tudi implementirat zaslon, ki je viden ko je okno zadosti široko. A screenshot of a computer

Description automatically generated

## Nadaljnji razvoj

Nadaljnji razvoj je imel velik poudarek na optimizaciji ter odpravljanju napak. Dodal sem tudi enostavnejšo različico prikaza spomina, ki se vključi v nastavitvah. Največji dodatek je bil uvedba disassembliranja ali razstavljanja. To omogoča uporabniku da iz strojne kode razstavi ukaze v assembly jezik. Ta funkcija je dostopna ko je način pisanja nastavljen na spomin(direktno pisanje v spomin). Ko se disassembler zažene bo uporabnik vprašan kje se program začne, to omogoča uporabo prvih pomnilniških naslovov za spremenljivke. Podatke pred tem bo disassembler zapisal z ukazom ».BYTE«. Če dissasembler naleti na neznan ali ne podpiran ukaz bo uporabnik vprašan kje naj se razstavljanje nadaljuje. Tudi tu bodo manjkajoči podatki zapisani z ukazom ».BYTE«.

Še en omembe vreden dodatek je uvedba možnosti da procesor deluje po hitrosti »ciklov« na sekundo. Tako deluje tudi pravi procesor. Vsak ukaz zahteva nekaj ciklov procesorja. Privzeta nastavitev je še vedno ukazi na sekundo, saj je tako enostavnejše upravljati z hitrostjo programa.

# Model

## Notranje komponente emulatorja.

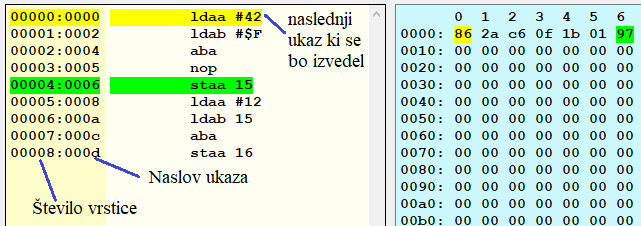
### Assembler(Sestavljalec)

Assembler je programska operma, ki prevaja ukaze, operacije, načine naslavlanja ter operande v njihove številske vrednosti(strojno kodo) določenega assembly jezika. Ta jezik je v primeru mojega emulatorja M6800 in M6803. Kot mnogi drugi assemblerji, uporablja in prepozna simbolične reference. To je način poimenovanja določenih pomnilniških lokacij ali ukazov, katere uporaba nadomesti ročne izračune in posodobitve naslovov po spremembah programa ter olajšajo uporabo spremenljivk. Moj assembler za M680X je »One-pass assembler«, to pomeni da gre skozi kodo samo enkrat in rezervira prostor za simbole, ki še niso bili definirani. Na koncu pa assembler preide skozi rezervirane pomnilniške naslove in jih nadomesti s vrednostjo nazivov.

## Uporabniški vmesnik

### Števec vrstic

Na levi strani glavnega okna je polje namenjeno štetju vrstic assembly programa. Vsaka vrstica tega polja ustreza številu vrstic polja za vpis assembly programa z indeksiranjem, ki se začne pri 0. Ko program trenutno ni sestavljen ali spomin ni razstavljen bodo v polju pisale samo vrsticam ustrezna števila. Če pa je program sestavljen oz. ukazi ustrezajo strojni kodi v spominu pa bo ob številu vrstic, ločeno s dvopičjem izpisan pomnilniški naslov, ki ustreza pomnilniški lokaciji kjer je zapisan prvi bajt ukaza oz. njegova operacijska koda. Takrat bo tudi vidna rumena označba, ki bo kazala na ukaz kateri se nahaja na trenutni vrednosti PC. To je tudi naslednji ukaz ki se bo izvedel.



V primeru da je vklopljena nastavitev »napredne informacije programa« pa bodo ob pomnilniški lokaciji bili izpisani vsi bajti v katere se ukaz na ustrezni vrstici sestavi ločeni z dvopičjem.

A close-up of a clock

Description automatically generated

Klik na vrstico tega polja bo pod pogojem, da je program sestavljen, obarval ukaz, število vrstice ter naslov prvega bajta tega ukaza v spominu z zeleno barvo. Ta oznaka bo ostala dokler se program ne spremeni ali uporabnik ne pritisne desni gumb miške na to polje.

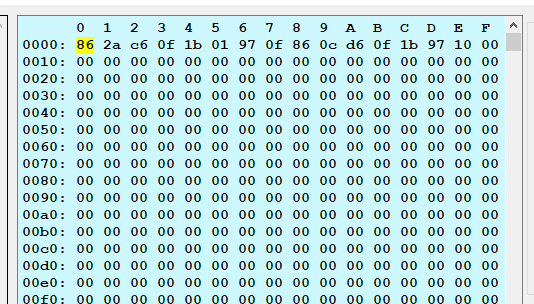
A screenshot of a computer

Description automatically generated

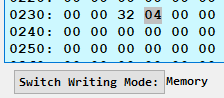
### Polje za vpis assembly kode

Ob števcu vrstic je polje za vpis assembly kode. Vpis v polje je mogoč, ko je emulator nastavljen na način vpisa kode.

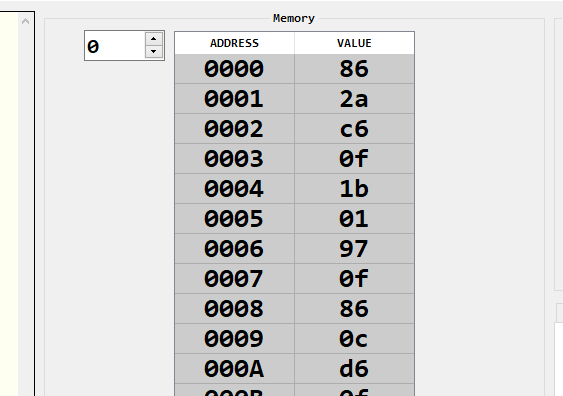
### Polje za prikaz spomina

Polje za prikaz spomina, ki je na sredini glavnega okna v podobi tabele prikazuje vse pomnilniške celice ter vrednosti ki jih hranijo v šestnajstiškem številskem sistemu. 

V načinu pisanja v spomin bo viden kazalec za spremembo pomnilniške lokacije. Uporabnik ga premika z puščicami na tipkovnici in vrednost celice zapiše z števkami šestnajstiškega številskega sistema.



Enostavni spomin je prikazan v obliki tabele. V levem stolpcu je pomnilniški naslov, v desnem pa vrednost spomina na tem naslovu. Prikazani naslovi se spreminjajo s števcem, ki je na levi strani tabele.



### Registri

Desno zgoraj je polje ki prikazuje vsebine registrov procesorja M6800, ti so akumulator A, akumulator B, indeksni register X, programski števec, kazalec sklada in zastavice stanja(half-carry, interrupt flag, negative, zero, overflow, carry). A screenshot of a computer

Description automatically generated

### Stran z večmi zavihki

Desno spodaj je stran za izbiro večih zavihkov.

1. A screenshot of a computer

   Description automatically generatedPrvi je okno konzole ki prikazuje napake, opozorila in informacije ki jih emulator posreduje uporabniku.
2. A screenshot of a computer

   Description automatically generatedDrugi vsebuje orodja za odpravljanje napak, ki vsebuje prevajalnik številskih sistemov ter sistem prelomnih točk ki omogočajo uporabniku samodejno ustavljanje izvajanja ukazov ob določenem dogodku ali pogoju. Na primer, program se lahko ustavi če po kateri register vseboval zaželeno vrednost, lahko se ustavi na določeni vrstici/ukazu ali pa, ko neka celica v spominu vsebuje zaželeno vrednost.
3. Tretji vsebuje večino nastavitev emulatorja. Možne nastavitve so:
   * Nastavitve, ki določajo kaj se bo prikazalo v oknu konzole.
   * Nastavitev, ki omogoča prikaz podrobnejših informacij vsakega sestavljenega ukaza.
   * Nastavitev, ki določa v kakšnem številskem sistemu bodo napisane vrednosti registrov.
   * Nastavitev, ki določa če bodo ukazi samodejno sestavljeni ko je program zagnan po spremembi ukazov.
   * Nastavitev, ki določa če bo se emulator samodejno ponastavil na stanje pred zadnjim izvrševanjem.
   * Nastavitev, ki določa, ali se procesor ravna po ciklični hitrosti ukazov in izvaja z določenim številom ciklov na sekundo ali pa sledi uporabnikovi nastavitvi ukazov na sekundo, pri čemer ciklična hitrost ni upoštevana.
   * Nastavitev, ki določa če bo uporabniku prikazan standardni prikaz spomina ki ima 16 pomnilniških naslovov in njihovih vrednosti v eni vrstici, ali enostavnejši prikaz ki prikazuje 20 poljubnih zaporednih pomnilniških naslovov in vrednosti katere vsebujejo v tabeli.
   * Nastavitev, ki omogoča nastavljanje delovanja emulatora med pisanjem kode ter neposredno v spomin, to posledično tudi določa če bo emulator kodo ali ukaze sestavljal ali razstavljal. Na spodnjem delu glavnega okna se pojavi gumb ki prikazuje trenutni način delovanja emulatorja ter ponuja uporabniku da ga zamenja. Privzeta nastavitev je da deluje na način sestavljanja, to je da uporabnik piše ukaze v assemblyu ter jih prevajalec sestavi v strojno kodo, druga nastavitev je način razstavljanja, takrat je uporabniku omogočeno spreminjanje strojne kode ter razstavljanje spomina, to je da se strojna koda v spominu prevede v assembly ukaze ki se bodo izpisali v polju za assembly program.
   * Nastavitvi za samodejno premikanje polja na ukaz kateri se trenutno izvršuje.
   * Nastavitev, ki določa kje bo prikazan zaslon. Če je nastavitev nastavljena na »glavno okno« bo zaslon prikazan na glavnem oknu med polju za spomin in polju registrov, ko bo ta imel zadosti prostora, oziroma ko bo glavno okno dovolj široko. Če je nastavitev nastavljena na »zunanje okno«, bo zaslon prikazan zunaj glavnega okna v obliki okna za dialog.
4. Četrti zavihek vsebuje opis in navodila za uporabo emulatorja.

A screenshot of a computer

Description automatically generated

1. A screenshot of a computer

   Description automatically generatedPeti vsebuje tabelo z vsemi ukazi mikroprocesorjev Motorola M6800 in Motorola M6803. Ukazi, operacijske kode, število ciklov, velikost ter opisi zadnje omenjenega so obarvani rdeče.

Na spodnji strani glavnega okna so gumbi ter izbirni meniji za olajšano in uporabniku prijaznejšo uporabo emulatorja. Ti so:

1. Gumb »Sestavi/Razstavi« bo če je način pisanja nastavljen na »pisanje kode« ukaze, ki so trenutno napisani v polju za pisanje ukazov sestavil in prevedel v strojno kodo, ki bo pregledna v polju za spomin. Če pa je način pisanja nastavljen na pisanje v spomin, bo ta gumb razstavil strojno kodo zapisano v spominu ter jo zapisal v polju za ukaze.
2. Meni za izbiro različice Motorola procesorjev. Trenutno emulator podpira dve različici Motorola M6800 in Motorola M6803, ampak je tako zasnovan da omogoča nadaljnjo razširitev ter dodajanje novih procesorjev. Emulator bo se ravnal, sestavljal razstavljal ter izvajal ukaze glede na izbran procesor.
3. Gumb za nalaganje, bo glede na izbiro načina pisanja iz zunanje datoteke naložil assembly kodo ali pa spomin.
4. Gumb za shranjevanje, bo glede na izbiro načina pisanja srahnil assembly kodo ali pa trenutni spomin v datoteko.
5. Gumb za zamenjavo načina delovanja emulatorja. Privzeto je ta gumb nedosegljiv lahko se pa prikaže s nastavitvijo »dovoli zamenjavo načina pisanja«. Desno od gumba je napis trenutnega načina.
6. Gumb za ponastavitev ponastavi emulator na stanje kakšnega je bil po zadnjem sestavljanju ali pred zadnjim izvajanjem.
7. Gumb »korak« ki izvede en ukaz z procesorjem izbranim v meniju za izbiro različice procesorja.
8. Gumb »Zaženi/Ustavi« v primeru, da procesor trenutno ne izvaja ukazov zažene samodejno izvajanje ukazov. V primeru, da procesor trenutno izvaja ukaze pa ga ta gumb začasno ustavi. Uporabnik lahko izvajanje vedno nadaljuje z vnovičnim pritiskom na ta gumb. Gumb deluje v skladu z menijem hitrosti ter nastavitvijo za izbiro načina izvajanja. Procesor bo samodejno prenehal izvajanje kadar naleti na strojni kod 0x00. Če je programski števec(PC) trenutno vsebuje pomnilniško lokacijo, ki vsebuje 0x00 in če je nastavitev »samodejna ponastavitev« vklopljena bo se emulator ponastavil.
9. Meni za nastavitev hitrosti izvajanja. Prikazuje število ukazov ali ciklov na sekundo. Možna je izbira od 1 do 1000 ter »maksimum«. Maksimum je včasih neenakomeren ampak je najhitrejša izbira. Te vrednosti so le približek saj je hitrost odvisna od zmogljivosti uporabnikovega računalnika.

# ZAKLJUČEK

Ta seminarska je malo podrobnejši vpogled v delovanje naprav USB in njihovih standardov. Na spletu res ni veliko seminarskih o USB v slovenščini še sploh pa ne tako podrobnih. V tej seminarski sem izpustil varnostne grožnje, kompatibilnost z drugimi standardi in protokole saj te stvari niso najpomembnejše po mojem mnenju.

Viri besedila

<https://en.wikipedia.org/wiki/Assembly_language#Assembler>

# VIRI

*Motorola 6800 - Wikipedia*. (11. 17 2023). Pridobljeno iz Wikipedia The Free Encyclopedia: https://en.wikipedia.org/wiki/Motorola\_6800

*QT Group*. (15. 11 2023). Pridobljeno iz spletnega mesta podjetja QT inc.: https://www.qt.io/

1. *Master/slave(Technology)* 2022 [online]. Wikipedija, The Free Encyclopedia. Pridobljeno 7. 3. 2022 na spletnem naslovu: https://en.wikipedia.org/wiki/Master/slave\_(technology)
2. *USB* 2022 [online]. Wikipedija, The Free Encyclopedia. Pridobljeno 7. 3. 2022 na spletnem naslovu: https://en.wikipedia.org/wiki/USB
3. *Universal Serial Bus Specification* 2022 [online]. Universal Serial Bus Specification. Pridobljeno 7. 3. 2022 na spletnem naslovu: https://fl.hw.cz/docs/usb/usb10doc.pdf
4. *USB Hardware* 2022 [online]. Wikipedija, The Free Encyclopedia. Pridobljeno 7. 3. 2022 na spletnem naslovu: https://en.wikipedia.org/wiki/USB\_hardware
5. *USB 3.0* 2022 [online]. Wikipedija, The Free Encyclopedia. Pridobljeno 7. 3. 2022 na spletnem naslovu: https://en.wikipedia.org/wiki/USB\_3.0
6. *USB-C* 2022 [online]. Wikipedija, The Free Encyclopedia. Pridobljeno 7. 3. 2022 na spletnem naslovu: https://en.wikipedia.org/wiki/USB-C
7. *Star network* 2022 [online]. Wikipedija, The Free Encyclopedia. Pridobljeno 7. 3. 2022 na spletnem naslovu: https://en.wikipedia.org/wiki/Star\_network
8. *USB(Communications)* 2022 [online]. Wikipedija, The Free Encyclopedia. Pridobljeno 7. 3. 2022 na spletnem naslovu: https://en.wikipedia.org/wiki/USB\_(Communications)
9. *Univerzalno serijsko vodilo* 2022 [online]. Wikipedija, prosta enciklopedija. Pridobljeno 7. 3. 2022 na spletnem naslovu: https://sl.wikipedia.org/wiki/Univerzalno\_serijsko\_vodilo
10. *What is Micro USB Pinout and Types* 2022 [online]. eTechnophiles. Pridobljeno 7. 3. 2022 na spletnem naslovu: https://www.etechnophiles.com/micro-usb-pinout-features/
11. *Univerzalno serijsko vodilo* 2022 [online]. Wikipedija, prosta enciklopedija. Pridobljeno 7. 3. 2022 na spletnem naslovu: https://sl.wikipedia.org/wiki/Univerzalno\_serijsko\_vodilo
12. *What is USB-C* 2022 [online]. electronicsnotes Pridobljeno 7. 3. 2022 na spletnem naslovu: https://www.electronics-notes.com/articles/connectivity/usb-universal-serial-bus/usb-c-connector.php
13. *USB 3.2 Specification Language Usage Guidelines from USB-IF* [online]. USB-IF Pridobljeno 7. 3. 2022 na spletnem naslovu: https://www.usb.org/sites/default/files/usb\_3\_2\_language\_product\_and\_packaging\_guidelines\_final.pdf

Viri slik

1. *USB* 2022 [online]. Wikipedija, The Free Encyclopedia. Pridobljeno 7. 3. 2022 na spletnem naslovu: https://en.wikipedia.org/wiki/USB
2. *What is Micro USB Pinout and Types* 2022 [online]. eTechnophiles. Pridobljeno 7. 3. 2022 na spletnem naslovu: https://www.etechnophiles.com/micro-usb-pinout-features/
3. *USB(Communications)* 2022 [online]. Wikipedija, The Free Encyclopedia. Pridobljeno 7. 3. 2022 na spletnem naslovu: https://en.wikipedia.org/wiki/USB\_(Communications)