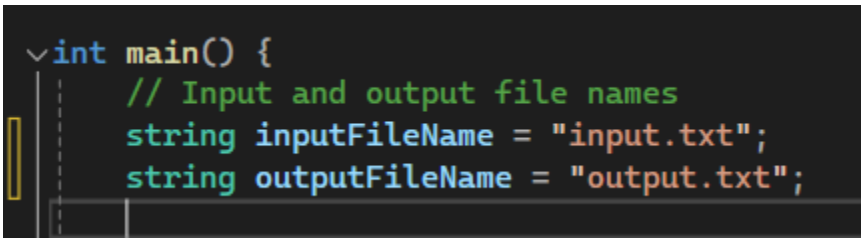
ესემბლი

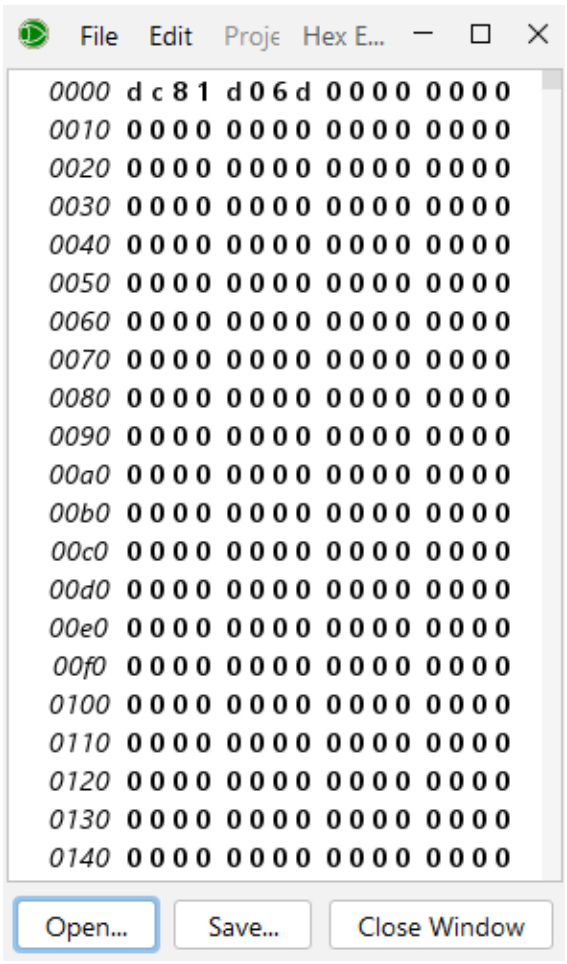
პროცესორზე კოდის გასაშვებად assembler ფაილში მოცემულია შესაბამისი კოდი main.cpp რომელიც იღებს ესემბლიში დაწერილ კოდს და აბრუნებს პროცესორისათვის გასაგებ სკრიპტს.

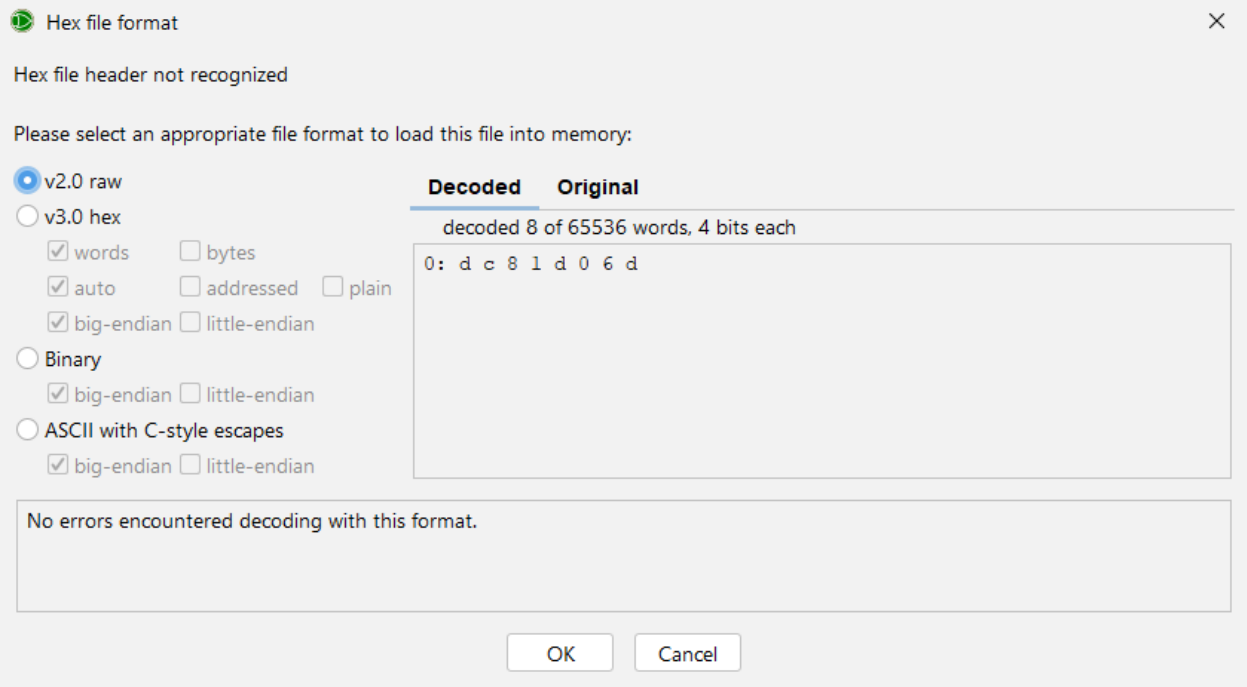
საჭირო ფაილის სათარგმნად კოდში უნდა მიუთითოს სათარგმნი ფაილის დასახელება იმ ფაილის დასახელება სადაც გინდათ რომ ნათარგმნი კოდი ჩაიწეროს.



იმისთვის რომ ესემბლერმა სწორად თარგმნოს კოდი აუცილებელია თითოეული ბრძანების თითო ხაზზე დაწერა (ბრძანებები განმარტებებით მოცემულია ცხრილში), სასვენი ნიშნებისა და ზედმეტი სიმბოლოების გარეშე.

ესემბლერის მიერ დაბრუნებული ფაილი უნდა ატვირთოთ პროცესორზე შეერთებულ რომში “open...” ღილაკით ხოლო ფაილის ფორმატში აირჩიოთ “v2.0 raw”.


File
Edit
Proje
Hex E...
0000 dc81 d06d 0000 0000
0010 0000 0000 0000 0000
0020 0000 0000 0000 0000
0030 0000 0000 0000 0000
0040 0000 0000 0000 0000
0050 0000 0000 0000 0000
0060 0000 0000 0000 0000
0070 0000 0000 0000 0000
0080 0000 0000 0000 0000
0090 0000 0000 0000 0000
00a0 0000 0000 0000 0000
00b0 0000 0000 0000 0000
00c0 0000 0000 0000 0000
00d0 0000 0000 0000 0000
00e0 0000 0000 0000 0000
00f0 0000 0000 0000 0000
0100 0000 0000 0000 0000
0110 0000 0000 0000 0000
0120 0000 0000 0000 0000
0130 0000 0000 0000 0000
0140 0000 0000 0000 0000
Open...
Save...
Close Window



ბრძანებების ცხრილი

#	ბრზანება	სინტაქსი
1	LOAD	(LOAD From to) from_პარამეტრი არის რეგისტრის ნომერი რომელშიც წერია ram-ის ის მისამართი საიდანაც გვინდა ინფორმაციის ჩამოტვირთვა, to_პარამეტრი არის რეგისტრის ნომერი რომელშიც გვინდა ინფორმაციის ჩამოტვირთვა.
2	STORE	(STORE From to) from_პარამეტრი არის რეგისტრის ნომერი საიდანაც გვინდა ინფორმაციის გადატანა ram-ში, to_პარამეტრი არის რეგისტრის ნომერი რომელშიც წერია ram-ის მისამართი სადაც გვინდა ინფორმაციის ჩაწერა.

3	ADD	(ADD A B TO) A და B _ პარამეტრები არის რეგისტრების ნომრები რომლებში არსებული ინფორმაციის შეკრებაც გვინდა, TO _ პარამეტრი არის რეგისტრის ნომერი სადაც გვინდა ჯამის შენახვა.
4	SUB	(SUB A B TO) A და B _ პარამეტრები არის რეგისტრების ნომრები რომლებში არსებული ინფორმაციის გამოკლებაც გვინდა, TO _ პარამეტრი არის რეგისტრის ნომერი სადაც გვინდა სხვაობის შენახვა.
5	MUL	(MUL A B TO) A და B _ პარამეტრები არის რეგისტრების ნომრები რომლებში არსებული ინფორმაციის გამრავლებაც გვინდა, TO _ პარამეტრი არის რეგისტრის ნომერი სადაც გვინდა ნამრავლის შენახვა.
6	SHIFTR	(SHIFTR P1 P2 TO) P1 _ პარამეტრი არის რეგისტრის ნომერი რომელში არსებული ინფორმაციიის ჩაჩოჩებაც გვინდა მარჯვნივ, P2 _ პარამეტრი არის რეგისტრის ნომერი რომელშიც წერია რამდენით გვინდა ჩაჩოჩება, TO _ პარამეტრი არის რეგისტრის ნომერი სადაც გვინდა პასუხის შენახვა.
7	DIV	(DIV A B TO) A და B _ პარამეტრები არის რეგისტრების ნომრები რომლებში არსებული ინფორმაციის გაყოფაც გვინდა, TO _ პარამეტრი არის რეგისტრის ნომერი სადაც გვინდა განაყოფის შენახვა.
8	SHIFTL	(SHIFTL P1 P2 TO) P1 _ პარამეტრი არის რეგისტრის ნომერი რომელში არსებული ინფორმაციიის ჩაჩოჩებაც გვინდა მარცხნივ, P2 _ პარამეტრი არის რეგისტრის ნომერი რომელშიც წერია რამდენით გვინდა ჩაჩოჩება, TO _ პარამეტრი არის რეგისტრის ნომერი სადაც გვინდა პასუხის შენახვა.
9	NOT	(NOT P TO) P _ პარამეტრი არის რეგისტრის ნომერი რომელში არსებული ინფორმაციაზეც გვინდა NOT ოპერაციის გამოძახება, TO _ პარამეტრი არის რეგისტრის ნომერი სადაც გვინდა პასუხის შენახვა.
10	AND	(AND A B TO) A და B _ პარამეტრები არის რეგისტრების ნომრები რომლებში არსებული ინფორმაციაზეც გვინდა AND ოპერაციის გამოძახება, TO _ პარამეტრი არის რეგისტრის ნომერი სადაც გვინდა პასუხის შენახვა.

11	OR	(OR A B TO) A და B _ პარამეტრები არის რეგისტრების ნომრები რომლებში არსებული ინფორმაციაზეც გვინდა OR ოპერაციის გამოძახება, TO _ პარამეტრი არის რეგისტრის ნომერი სადაც გვინდა პასუხის შენახვა.
12	XOR	(XOR A B TO) A და B _ პარამეტრები არის რეგისტრების ნომრები რომლებში არსებული ინფორმაციაზეც გვინდა XOR ოპერაციის გამოძახება, TO _ პარამეტრი არის რეგისტრის ნომერი სადაც გვინდა პასუხის შენახვა.
13	NAND	(NAND A B TO) A და B _ პარამეტრები არის რეგისტრების ნომრები რომლებში არსებული ინფორმაციაზეც გვინდა NAND ოპერაციის გამოძახება, TO _ პარამეტრი არის რეგისტრის ნომერი სადაც გვინდა პასუხის შენახვა.
14	NOR	(NOR A B TO) A და B _ პარამეტრები არის რეგისტრების ნომრები რომლებში არსებული ინფორმაციაზეც გვინდა NOR ოპერაციის გამოძახება, TO _ პარამეტრი არის რეგისტრის ნომერი სადაც გვინდა პასუხის შენახვა.
15	JUMP	(JUMP BR) BR _ პარამეტრი არის რეგისტრის ნომერი სადაც წერია კოდის მერამდენე ხაზზე გვინდა გადახტომა.
16	LI	(LI TO INT) TO _ პარამეტრი არის რეგისტრის ნომერი სადაც გვინდა ინფორმაციის შენახვა, INT _ არის 0-დან 255-ამდე ხელით შეყვანილი რიცხვი რომლის შენახვაც გვინდა.
17	BREQ	(BRNE P1 P2 BR) P1 და P2 _ პარამეტრები არის რეგისტრების ნომრები რომლებში არსებული ინფორმაციების შედარებაც გვინდა, თუ ისინი უდრის ერთმანეთს მოხდება გადახტომა, BR _ პარამეტრი არის რეგისტრის ნომერი სადაც წერია კოდის მერამდენე ხაზზე გვინდა გადახტომა.
18	BRNE	(BRNE P1 P2 BR) P1 და P2 _ პარამეტრები არის რეგისტრების ნომრები რომლებში არსებული ინფორმაციების შედარებაც გვინდა, თუ ისინი არ უდრის ერთმანეთს მოხდება გადახტომა, BR _ პარამეტრი არის რეგისტრის ნომერი სადაც წერია კოდის მერამდენე ხაზზე გვინდა გადახტომა.

ისეთი დამატებითი ფუნქციონალის გამოყენება როგორიცაა ტაიმერი და GPIO ხდება რეგისტრებში ინფორმაციის ჩაწერა ამოკითხვის საშუალებით.

ჯამში გვაქვს 16 ხელმისაწვდომი რეგისტრი თუმცა დამატებითი ფუნქციონალის გამო ყველას თავისუფალი გამოყენება შეზღუდულია.

ვინაიდან ლოგოსიმს აქვს მხოლოდ ცალმხრივი მიმოცვლა 8ბიტიანი GPIO-ს 4 ბიტი წარმოადგენს input-ს და 4 output-ს.

რეგისტრები

რეგისტრის ნომერი	დანიშნულება
0	OCA რეგისტრი _ ამ რეგისტრში რიცხვის ჩაწერის შემთხვევაში ტაიმერი დაყენდება ამ რეგისტრის მნიშვნელობაზე და როცა მიარწევს ამ მნიშვნელობას მისცემს სიგნალს IFOCA რეგისტრს.
1	OCB რეგისტრი _ ამ რეგისტრში რიცხვის ჩაწერის შემთხვევაში ტაიმერი დაყენდება ამ რეგისტრის მნიშვნელობაზე და როცა მიარწევს ამ მნიშვნელობას მისცემს სიგნალს IFOCB რეგისტრს.
2-5	ინფორმაციის გამომტანი რეგისტრები _ თუ ამ რეგისტრებში ჩაწერთ რიცხვს 1 შესაბამის პინზე გვექნება ლოგიკური მაღალი, ნებისმიერი სხვა მნიშვნელობისთვის კი ლოგიკური დაბალი.
6-9	ინფორმაციის წამკითხველი პინები _ ეს რეგისტრები ჩანაცვლებულია პირდაპირი ინფუთის პინებით ამიტომ მათშში რაიმეს ჩაწერა შეუძლებელია, შესაძლებელია მხოლოდ ინფორმაციის წამოღება, 1 ნიშნავს ლოგიკურ მაღალს ხოლო 0

	ლოგიკურ დაბალს, სხვა რიცხვით მნიშვნელობებს ისინი არ იძლევიან.
10	IFO რეგისტრი _ ამ რეგისტრში რიცხვის ჩაწერის შემთხვევაში როცა ტაიმერი მიაღწევს მაქსიმალურ მნიშვნელობას კოდში მოხდება ნახტომი ამ რეგისტრში გაწერილ ხაზზე.
11	IFOCA რეგისტრი _ თუ ტაიმერმა მიაღწია OCA მნიშვნელობას კოდში მოხდება ნახტომი ამ რეგისტრში გაწერილ ხაზზე. (თუ OCA რეგისტრი ცარიელია შესაძლებელია ამ რეგისტრის თავისუფლად გამოყენება)
12	IFOCB რეგისტრი _ თუ ტაიმერმა მიაღწია OCB მნიშვნელობას კოდში მოხდება ნახტომი ამ რეგისტრში გაწერილ ხაზზე. (თუ OCB რეგისტრი ცარიელია შესაძლებელია ამ რეგისტრის თავისუფლად გამოყენება)
13	ფინალური რეგისტრი _ თუ ამ რეგისტრში ჩაწერთ რიცხვს 1 ეს ნიშნავს კოდის დაასასრულს და მთვლელი აღარ გაიზრდება, ნებისმიერი სხვა მნიშვნელობისათვის ეს რეგისტრი იმუშავებს როგორც ხცვები.
14-15	თავისუფალი გამოყენების რეგისტრები _ ამ რეგისტრებს არ აქვთ რაიმე კონკრეტული დანიშნულება შესაძლებელია მათი კოდიში თავისუფლად გამოყენება.

ტესტები

1. მარტივი ციკლი _ რეგისტრის მნიშვენობა იზრდება ერთით სანამ არ მიაღწევს 60-ს შემდეგ კი ოპერაციულ მეხსიერებაში გამოიტანება მისამართზე “1” (16ობითში 60 არის 3c)
   simple loop _ assembly.txt
   simple loop.txt
2. ლედების ტესტი _ ციკლურად ირთვება და ითიშება ლედები.
   LED _ assembly.txt
   LED.txt
3. ტაიმერის ტესტი _ კოდი ტაიმერი ისეტება მეშვიდე ხაზზე, შემდეგ ასრულებს ოპერაციულ სისტემაში პასუხის გამოტანის ოპერაციას და უბრუნდება მეშვიდე ხაზს რითაც კოდი სრულდება (პასუხი არის 100 16ობითში 64 და გამოიტანება ნომერ 5 მისამართზე).
   timer _ assembly.txt
   timer.txt

მოცემული ტესტები მოთავსებულია ასემბლერის ფაილში.

განსაზღვრული ინსტრუქციების გამოყენებით შეგიძლიათ დაამატოთ სხვა ტესტებიც.