

Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана

(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.01 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

ОТЧЕТ

по лабораторной работе № 1

Дисциплина: Организация ЭВМ и систем

Преподаватель		_
	(Подпись, дата)	(И.О. Фамилия)
Студент гр. ИУ6-72Б		И.С. Марчук
1	(Подпись, дата)	— (И.О. Фамилия)

Задание 1

Собрать программу, перевести её в псевдокод. Код на языке ассемблера:

```
.section .text
     .globl _start;
len = 9 #Размер массива
enroll = 2 #Количество обрабатываемых элементов за одну итерацию
elem sz = 4 #Размер одного элемента массива
_start:
     la x1, _x
     addi x20, x0, (len-1)/enroll
     1 \text{w x} 31, 0(\text{x} 1)
     addi x1, x1, elem_sz*1
lp: 1w x2, 0(x1)
     1w x3, 4(x1)
     bltu x2, x31, lt1
     add x31, x0, x2 #!
lt1: bltu x3, x31, lt2
     add x31, x0, x3
lt2: add x1, x1, elem_sz*enroll
      addi x20, x20, - 1
     bne x20, x0, lp
lp2: j lp2
     .section .data
_x: .4byte 0x1
     .4byte 0x2
     .4byte 0x3
     .4byte 0x4
     .4byte 0x5
     .4byte 0x6
     .4byte 0x7
     .4byte 0x8
     .4byte 0x9
```

```
Код на языке Си:
#include <stdio.h>
#define len 9
#define enroll 2
#define elem sz 4
int _x[] = \{1,2,3,4,5,6,7,8,9\};
int x0 = 0;
int main(){
     int *x1 = _x;
     int x20 = (len - 1)/enroll;
     int x31 = x1[0];
                                          ..., x2 = 0, x3 = 0;
     *x1 = *x1 + (elem sz*1);
    while(x20 != 0){
          x2 = x1[0];
          x3 = x1[1];
          if (x2 > x31){
              if (!(x3 > x31)){
                     x31 = x3; // если x3 < x31
               }
          }else{
                x31 = x2; // если x2 < x31
         *x1 = *x1 + (elem_sz * enroll);
         x20 = x20 - 1;
    }
    printf("%d", x31);// поиск наименьшего
    return 0;
}
В конце выполнения программы в регистре х31 будет значение 1.
```

Код дизассемблера:

80000000 <_start>:

80000000: 00000097 auipc x1,0x0

80000004: 03c08093 addi x1,x1,60 # 8000003c <_x>

80000008: 00400a13 addi x20,x0,4 8000000c: 0000af83 lw x31,0(x1) 80000010: 00408093 addi x1,x1,4

80000014 <lp>:

80000014: 0000a103 lw x2,0(x1) 80000018: 0040a183 lw x3,4(x1)

8000001c: 01f16463 bltu x2,x31,80000024 < lt1>

80000020: 00200fb3 add x31,x0,x2

80000024 <lt1>:

80000024: 01f1e463 bltu x3,x31,8000002c <1t2>

80000028: 00300fb3 add x31,x0,x3

8000002c <lt2>:

8000002c: 00808093 addi x1,x1,8 80000030: fffa0a13 addi x20,x20,-1

80000034: fe0a10e3 bne x20,x0,80000014 <lp>

80000038 < lp2>:

80000038: 0000006f jal x0,80000038 < lp2>

Задание 2

Получить снимок экрана, содержащий временную диаграмму выполнения стадий выборки (итерация 2) и диспетчеризации команды с указанным адресом (80000028).

e - Default	Msgs						
/tb/ext_reset	0						
/tb/sys_clk /tb/cyc_ent	34	30	133	34		135	136
/tb/uut/cpu/bp/predicted_pc /tb/uut/cpu/bp/use_prediction	00000000	00000000					
. /tb/uut/cpu/gc_unit_block/gc_fetch_flush . /tb/uut/cpu/gc_unit_block/gc_fetch_hold	l ^o						
/tb/uut/cpu/instruction_bram/addr /tb/uut/cpu/fetch_block/pc	2000000a 80000028	20000008	[20000009 [80000024	[2000000a [80000028		2000000b 8000002c	12000000c 120000030
		3000020	10000021	10000020		200000021	18000030
/tb/uut/cpu/instruction_bram/data_out /tb/uut/cpu/fetch_block/fetch_instruction	00c0a103 00c0a103	0080a103	002f8fb3	00c0a103		002f8fb3	01008093
. /tb/uut/cpu/fetch_block/fetch_complete	1	0080a103	002f8fb3	[00c0a103		002f8fb3	X01008093
spatch	6	4	Į5	16		17	10
		4	[5	6		7	.0
/tb/uut/cpu/id_block/pc_id_next /tb/uut/cpu/id_block/pc_id_available		5	6	7		10	11
/tb/uut/cpu/id_block/pc_table	80000010 8000001	80000010 800	\$0000010 80000014 80000018 8000001c	8 [80000010 8000001	80000018 8000001c 8	80000010 80000014 80000018 8000001c 8	. (80000010 80000014 80000018 8000001
- [0]	80000010	80000010	E.				
→ [1]	80000014	80000014					
→ [2]	80000018	80000018					<u> </u>
	8000001c	8000001c					
	80000020	80000050	[80000020				
- ♦ [5]	80000024	80000034		80000024			
→ [6]	80000038	80000038				[80000028	
→ [7]	8000000c	8000000c					[8000002c
/tb/uut/cpu/id_block/instruction_table	002f8fb3 0040a103	Section of the sectio	100268653.0040a103.00268653.0080a103.66	f 002f8fb3 0040a103	D0266653.0060a103.00	1002f8fb3 0040a103 002f8fb3 0080a103 00	
√ [0]	002f8fb3	002f8fb3	2002 0103 00 108103 00210 E3 00008103 11	7 2002/0103 00108103	502 0155 0000a165 00	2002 0000 00100100 002100 00	200210103 00100103 00210103 00000103
-♦ [1]	0040a103	0040a103					
	002f8fb3	002f8fb3					
- ♦ [3]	0080a103	00000004	0080a103				
→ [4]	002f8fb3	fffa0a13		002f8fb3			
	fc0a1ce3	fc0a1ce3				100c0a103	·
-∜ [6]	001f8f93	001f8f93					1002f8fb3
→ [7]	0000a103	0000a103					
ecode & Issue /tb/uut/cpu/id_block/decode	2 80000018 002f8f	1.00000014.004	100.11	0.00000010.00000		Va 0000001 - 0000 - 100 1 1	
/m/and-thaila_biocklaecode	2 80000018 002/8/	1 80000014 0040	a) 03 1	2 80000018 002f8ft	7.1	3 8000001c 0080a103 1 1	

Рисунок 1 — Результат выполнения задания 2.

Сигнал $pc_{id_available}$ равен 1, что подтверждает готовность блока управления метаданными принять результат выборки. $pc=0\times80000000$. Адрес, соответствующий данному значению, а именно 0×2000000 выставляется на ША (сигнал addr).

Выставление сигнала en paspeшaeт paботу памяти команд. Одновременно с этим выставляется сигнал pc_id_assigned, указывающий блоку управления метаинформацией, что запрос в память отправлен, и информация о текущем pc должна быть записана в очередь команд. Текущее значение pc выдается в блоку управления метаинформацией через сигнал if pc. В начале следующего такта это значение (80000028) будет записано в

pc_table по индексу, равному значению текущего первого свободного id (в нашем случае 6), а pc_id увеличится на 1. Это означает, что команде, которая будет выбрана по адресу 8000000, присвоен id=0

В следующем такте (2) данные по адресу 80000028 (00с0а103) выставляются памятью команда (сигнал data_out). Эти данные выдаются из блока выборки (сигнал fetch_instruction), при этом сигнал fetch_complete равен 1. Операция выборки данной команды завершается в такте 34, однако значение данных подается на вход блока управления метаинформацией только в такте 35.

По фронту, завершающему такт 35, выбраный код команды (то есть, сигнал fetch_instruction) записывается в таблицу instruction_table по индексу 5. Это завершает операцию диспетчеризации.

Задание 3

Получить снимок экрана, содержащий временную диаграмму выполнения стадии декодирования и планирования (итерация 2) на выполнение команды с указанным адресом(80000034).

| The control of the

Снимок экрана представлен рисунком 2.

Рисунок 2 — Результат выполнения задания 3.

B такте 48 issue.pc = 80000034, issue.stage_valid = 1,

unit_issue[0].new_request = 1, значит в такте 47 команда декодировалась и запланировалась на выполнение.

Задание 4

Получить снимок экрана, содержащий временную диаграмму выполнения стадии выполнения команды(итерация 2) с указанным адресом(80000020).

CHUMOK SKPAHA ПРЕДСТАВЛЕН РИСУНКОМ 3.

| Warner Collade | Section | Section

Рисунок 3 — Результат выполнения задания 4.

Для блока АЛУ признаком, служащим для подтверждения готовности результата является сочетание сигналов unit_wb[0].done и unit_issue[0]/new_request_такое произошло в такте 41.

Задание 5

Получить снимок экрана, содержащий временные диаграммы сигналов, соответствующих всем стадиям выполнения команды, обозначенной в тексте программы символом #!

Команда - add x31, x0, x2

Дизассемблер: 80000020: 00200fb3 add x31,x0,x2

Снимок экрана представлен рисунком 4.

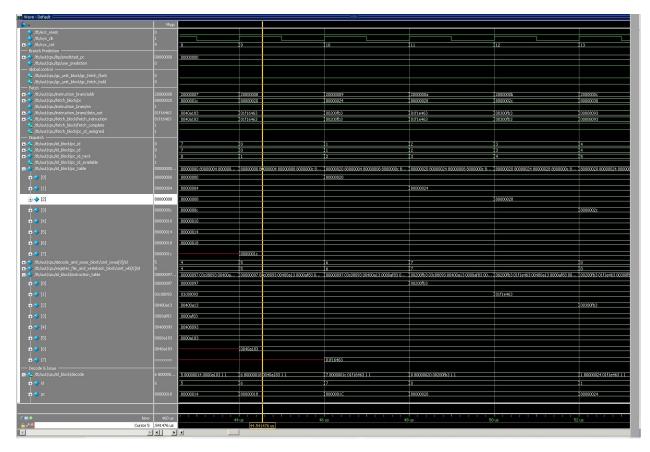


Рисунок 4 — Результат выполнения задания 5.

Номер такта	Действие
9	fetch
10	dispatch
11	Non decod
12	decod
13	issue

Анализируя диаграмму заполнить трассу выполнения программы.

Рекомендуется использовать для этого файл pipeline.ods, содержащий трассу тестового примера.

Результат представлен на рисунке 6.

020000	de constitución de la constituci	2000	1.1																									p Tar																										
Адрес	Код воманды	Команда		2 3		5 6	7	8 9	10 1	1 12	13 1	14 1	16	17 1	8 19	20	21 2	2 23	24 :	25 2	26 2	7 28	29	30	31 3					37 3	8 3	9 40	41	42 43	3 44	45	46 E	48	49	50 5	1 52	53	54	55 5	57	58	59 6	0 6	62	63	64	65	66	67
80000000 (start>	00000097	auipc x1,0x0	0 E	ID I	AL																	3													3 1 1						8						1							
80000004	03c08093	addi x1,x1,60 # 8000003c <_x>	1	rI							_	_		_	_		_	_		_	_	_	_		_	_	_	ш		_		_	ш	_	_		_		_	_	_	_		_	_		_	_	_			_	-	
80000008	00400a13	addi x20,x0,4	2	1	ID					1 5							- 15					1	-		- 13												-						100				-	1				1200		
8000000c 80000010	0000af83	lw x31,0(x1) addi x1,x1,4	3				M1 N				_			_	_	_	_	_		_	_	_	_		_	-	_	_	_	_	_	-	ш	_	_	\perp	_	_	_	_	_	-		_	_		_	_	-		_	_	-	
	00408092	addi xi,xi,4	4																				-				-					-															-							
80000014 <1p> 80000018	0040a103	lw x2,0(x1) lw x2,4(x1)	5		_	1	ID		M2 M					_	_		_	_		_	-	_	-	_	_	-	-	-	_	_	_	-	_	_	-	_	_	_	_	_	_	_		_	_		_	_	-		_	_	-	_
80000018		lw x3,4(x1) bleu x2,x31,80000024 <1e1>	0						D I						-						-	1	-		-	-	-			-	4	-					-										-		-					
8000001c 80000020	00200fb2	add m31, m0, m2	7		_	_			ID I			_	-	_	_	-	_	-		_	-	-	-	_	_	-	-	-	_	_	-	-	_	_	-	_	_	-	_	_	-	-	_	_	-	_	_	-	-	_	_	_	-	_
80000020	01f1e463	hltu x3,x31,8000002c <1t2>	0					2		D W					-								-		-	-	-			-	-	-					-			-				-			-							
80000024 <161>	00300fb3	add m31, m0, m3	1	_	-	_	_	_		ID				_	-	-	_	-		_	-	-	-	_	_	-	-	-	_	-	-	-	_	_	-	_	-	-	_	-	-	-	_	-	-	_	-	-	-	_	_	_	_	_
80000028 8000002c <1t2>	00300263	addi x1,x1,0	2						- 2		ID						- 0				-		-		-		-					-															-							
80000020 <162>	fffa0a13	addi x20.x201	3		-		_	_		E			D		-	-		-		-	-		_	_	-	-	-	-	_	-	-	-	-		-	_	-	-	_	-	-	-		-			-	-			_		_	_
80000034	fe0al0e3	bne x20,x0,80000014 <1p>	4										8										-				-					-															-							
80000028 <1p2>	00000006f	isl #0,80000028 <1m2>	0		-	-	_	_		-			ID				-	-		-	-	-	-	_	-	-	-	-	_	-	-	-	-	_	-	_	-	-	_	-	-	-	_	-	-		-	-	-		_	_	\vdash	_
8000002c <1p2>	000000001	<pre><invalid operation=""></invalid></pre>	0									-		ID I							-	9	-		-	-	-					-						-									-							
80000040	000000002	<pre><invalid operation=""></invalid></pre>	- 6											FI									-			-	-		_	-	-	-			-				_			-					_	-						_
80000044	000000002	<pre><invalid operation=""></invalid></pre>	0												FE					-	-		-		-	-	-		-	-	-	-		-			-	-		-	-			-			-							
80000044	000000004	<pre><invalid operation=""></invalid></pre>	14												F 2.																	1																	-					
80000048 80000014 <1p>	0000a103	<pre>%invalid operation> lw x2.0(x1)</pre>	2												22		TD T	M	M2 1	M2			-																															-
80000018	0040a183	lw #3,4(#1)	-											-					MI I		**		-	-		-			_	-						_											-	-						
8000001c	01f16463	bltu #3,#31,80000024 <1t1>	0																D				-			-	-			-		-								-	-						-							
80000020	00200fb3	add x31,x0,x2	2		-		_							-	-				ID				-	-	-	-	•		-	-	-		-			_	-					-	_	-			-	-					-	_
80000024 <1t1>	01f1e463	bltu x3,x21,8000002c <1t2>	2												-			S.A.		ID I			-			-	-			-	-	-			-					-	-						-							
80000028	00200fb2	add x21,x0,x2												-								D	NT.			-			-	-	-		-			_											-							
8000002c	00808093	addi xl,xl,8													_							D W		λL	-	-	-	_		_	-	-		-	-		-			-	_			_	-		-		-					
80000020	fffa0a12	addi x20,x20,-1																				ID			37		-		-	-																	-					1000		
80000034	fe0al0e3	bne x20,x0,80000014 <1p>	2												_					-	-				D I		-	_	_	_	-	-		-	_		-			-	_			_	-		-		-					
80000014 <1p>	0000a103	h x2.0(x1)	0																					ID			1 H2	142																								1000		
80000018	0040a183	lw x3.4(x1)	1																		-			r				H2	M2		-	-		_	_		_			_	_						-		-					
8000001c	01f16463	bltu x2,x31,80000024 <1t1>	2																1000			2						C																	1000							2000		
80000020	00200fb3	add x31,x0,x2	2																				-					W		AT.	_																							
80000024 <1t1>		bltu x3,x31,8000002c <1t2>	4																1000									8		D I	R								100						1000							2000		
80000028	00200fb2	add x31,x0,x3	- 5																		-		-		_	-		ID							_		_			_	_						-		_					
8000002c <1t2>	00808092	addi ml,ml,8	6																								-			2		AL													1000							2000		
80000020	fffa0a13	addi x20,x20,-1	2																				_			-		-		ID I	2 2									_							-		_					
80000024	fe0al0e3	bne #20, #0, 80000014 <1m>	2								0000											C Common						1000		E I			C	B											1000							10000		
80000014 <lp></lp>	0000a103	lw x2.0(x1)	4																		_		_		_	_	_							D H	1 M2	M2	_			_	_			_			-		_					
80000018	0040a183	lw x3,4(x1)	5														- 60		1000			C C						de la const						WI			42								1000							2000		
8000001c		bltu x2,x31,80000024 <1t1>	6																				_			_				_	-			W W													_							
80000020	00200fb3	add x31,x0,x2	4			4				4							- 1		1200			S C						Sec.				i i		ID W				9	200		S.		100		1000							2000		
80000024 <1t1>	01f1e463	bltu x3,x31,8000002c <1t2>	5																				1											8 11															П					
80000028	00200fb2	add x21,x0,x2	6			10											15																	E	ID		8 5		λL						100				i.			1000		
8000002c <1t2>	00808092	addi xl.xl.8	7																				-1											-				8		AL														
80000020	fffa0a13	addi #20,#20,-1	0														-																			E	ID 9	W	W	D A	L								i.			100		
80000024	fe0#10e3	bne x20,x0,80000014 <1p>	1																				1															ID		WE														
80000014 <1p>	0000a103	lw x2,0(x1)	2			10											10											land.							S .					WV									i.			100		
80000018	0040a183	ly x3.4(x1)	3																				-1															-		ID V														
8000001c	01f16463	bltu #2,#31,80000024 <1t1>	4			6				4		6							1300 (2		- 0				- 6			Day.			6				C C		64	100		E 10				- 0	120							2000		
80000020	00200fb3	add x31,x0,x2	5																				1			т														F	ID						1		Т					
80000024 <1t1>	01f1e463	bltu x3,x31,8000002c <1t2>	2			15											15		No.			1						la se		- 1					1		43					X			100							120		
80000028	00200fb2	add x31,x0,x3	3																				_																			FX												
80000028 <1p2>	0000006f	ial #0,80000028 <1p2>	4			- 1				4		26					- 13		100			1			-			200							4		100	100	200				2	ID I	AL							1000		
8000002c	00000001	<pre><invalid operation=""></invalid></pre>	5																				_									1													D									
80000040	000000002	<pre><invalid operation=""></invalid></pre>	6								000						- 65		1000		- 2	1						100							S S		64		200		S.				ID							2000		
80000044	00000000	<invalid operation=""></invalid>	4																																										F	X								
80000048	00000004	<invalid operation=""></invalid>	5			100				1							- 6		1200		- 2	1													S Company		64		200		8			- 6	120	FX						1000		
80000038 <1p2>	0000006£	jal x0,80000038 <1p2>	6																				1																								E 1	D E	В					
Carolina .	200000000000000000000000000000000000000	Комания	1.1	2 3	1	5 6	7	8 9	10 1	1 12	13 1	14 1	16	17 1	8 19	20	21 2	2 23	24 :	25 2	26 2	7 28	29	30	31 3	2 3	3 34	35	36	37 3	8 3	9 40	41	12 1	3 44	45	16 1	48	49	50 5	1 52	53	54	55 5	57	58	59 6	0 6	62	63	64	65	66	67
Ampec	Код команды	Команда	10		_		_		_		_		_	_		_	_	_		_	_	_	_	-	_		House			_		_	_	_	_	_		_	_		_	_	_		_	_	_		_	_	_	_	_	_

Рисунок 6 — Диаграмма неоптимизированного кода

По трассе видно наличие множественных гз конфликтов, в момент декодирования. Конфликты возникают при декодировании команд из блока АЛУ или из блока обращения к памяти после декодирования команд обращения к памяти. Повысить эффективность программы можно двумя способами: уменьшить количество конфликтных ситуаций или уменьшить количество ветвлений.

Конфликты возникают при декодировании команд из блока АЛУ или из блока обращения к памяти после декодирования команд обращения к памяти. Провести оптимизацию программы путем перестановки команд для устранения конфликтов.

Код на ассмеблере:

```
.section .text
```

.globl _start; len = 9

#Размер массива

enroll = 2 #Количество обрабатываемых элементов за одну итерацию elem sz = 4 #Размер одного элемента массива

```
_start:
    la x1, _x
    addi x20, x0, (len-1)/enroll
lw x31, 0(x1)    addi x1, x1,
```

```
elem_sz*1 lp:
                    1w x2, 0(x1)
1w x3, 4(x1)
                  addi x20,
x20, - 1
     bltu x2, x31, lt1
add x31, x0, x2 #! lt1:
bltu x3, x31, lt2
add x31, x0, x3 lt2:
     add x1, x1, elem_sz*enroll
bne x20, x0, lp lp2: j lp2
     .section .data
      .4byte 0x1
_x:
     .4byte 0x2
     .4byte 0x3
     .4byte 0x4
     .4byte 0x5
     .4byte 0x6
     .4byte 0x7
     .4byte 0x8
     .4byte 0x9
Код на языке Си:
#include <stdio.h>
#define len 9
#define enroll 2 #define
elem_sz 4 int _x[] =
\{1,2,3,4,5,6,7,8,9\}; int x0 =
0;
int main()
      int *x1 = _x; int x20 = x0 +
       (len - 1)/enroll; int x31 =
      x1[0], x2 = 0, x3 = 0; *x1 =
       *x1 + (elem_sz*1); while(x20
       != x0)
       \{ x2 = x1[0]; x3 \}
             = x1[1];
             x20 = x20 - 1;
             if (x2 > x31)
```

```
if (x3 > x31)
                 }
                 else
                 \{ x31 = x0 + x3; 
           }
           else
           \{ x31 = x0 + x2; 
        x_1 = x_1 + (elem_sz * enroll);
  printf("%d", x31);
return 0;
}
Код дизассемблера:
80000000 <_start>:
80000000:
                               auipc x1,0x0
             00000097
                               addi x1,x1,60 # 8000003c < x>
80000004:
             03c08093
80000008:
             00400a13
                               addi x20,x0,4
800000c:
             0000af83
                               1w
                                     x31,0(x1)
80000010:
             00408093
                                addi x1,x1,4
80000014 <lp>:
80000014:
             0000a103
                               lw
                                     x2,0(x1)
80000018:
                               1w
                                     x3,4(x1)
             0040a183
8000001c:
                              addi x20,x20,-1
             fffa0a13
                               bltu x2,x31,80000028 < lt1>
80000020:
             01f16463
             00200fb3
                               add
                                     x31,x0,x2
80000024:
80000028 <lt1>:
80000028:
             01f1e463
                               bltu
                                     x3,x31,80000030 < lt2>
8000002c:
             00300fb3
                               add
                                     x31,x0,x3
80000030 <lt2>:
80000030:
             00808093
                               addi x1,x1,8
                                     x20,x0,80000014 <lp>
80000034:
             fe0a10e3
                               bne
80000038 <lp2>:
80000038:
             000006f
                               jal
                                    x0,80000038 < lp2>
```

Составим новую трассу выполнения.

Снимок экрана представлен рисунком 7.

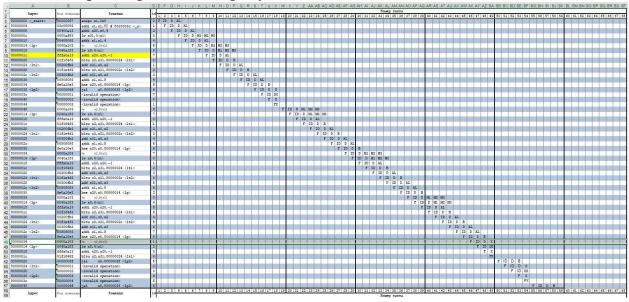


Рисунок 7 — Результат выполнения задания 4.

Вывод: в ходе выполнения лабораторной работы были изучены принципы функционирования, построения и особенности архитектуры супер скалярных конвейерных микропроцессоров, принципы проектирования и верификации сложных цифровых устройств с использованием языка описания аппаратуры SystemVerilog и ПЛИС. Проанализирована и рассмотрены способы улучшения программы на языке Assembler с помощью ядра Taiga, развернутого под САПР Intel Quartus.