

Preparation Report LAB3

Guy Cohen 207881004

Liav Ben Or 315909390

מבוא

במעבדה זו מימשנו מעבד Multy Cycle המורכב מ Data Path ו control unit. המערכת תדע לממש את מספר ההוראות המצורפות ב isa הנתונה:

Instruction Format	Decimal value	OPC	Instruction	Explanation	N	Z	C
R-Type	0	0000	add ra,rb,rc	$R[ra] \leq R[rb] + R[rc]$	*	*	*
			nop	$R[0] \leq R[0] + R[0]$ (<i>emulated instruction</i>)	*	*	*
	1	0001	sub ra,rb,rc	$R[ra] \leq R[rb] - R[rc]$	*	*	*
	2	0010	and ra,rb,rc	$R[ra] \leq R[rb] \text{ and } R[rc]$	*	*	-
	3	0011	or ra,rb,rc	$R[ra] \leq R[rb] \text{ or } R[rc]$	*	*	-
	4	0100	xor ra,rb,rc	$R[ra] \leq R[rb] \text{ xor } R[rc]$	*	*	-
	5	0101	<i>unused</i>				
J-Type	6	0110	<i>unused</i>				
	7	0111	jmp offset_addr	$PC \leq PC + 1 + \text{offset_addr}$	-	-	-
	8	1000	jc /jhs offset_addr	If(Cflag==1) $PC \leq PC + 1 + \text{offset_addr}$	-	-	-
	9	1001	jnc /jlo offset_addr	If(Cflag==0) $PC \leq PC + 1 + \text{offset_addr}$	-	-	-
	10	1010	<i>unused</i>				
I-Type	11	1011	<i>unused</i>				
	12	1100	mov ra,imm	$R[ra] \leq \text{imm}$	-	-	-
	13	1101	ld ra,imm(rb)	$R[ra] \leq M[\text{imm} + R[rb]]$	-	-	-
	14	1110	st ra,imm(rb)	$M[\text{imm} + R[rb]] \leq R[ra]$	-	-	-
	15	1111	done	Signals the TB to read the DTCM content	-	-	-

Note: * The status flag bit is affected , - The status flag bit is not affected

Table 1 : Multi-cycle CPU ISA

ביצענו את חלק ה control ע"י מימוש דיאגרמת מצבים מסוג Mealy FSM, בעזרת קווי הבקרה היוצאים מיחידת ה control, יחידת ה datapath יכולה לתמוך בהוראות שב ISA

3. Controller based system:

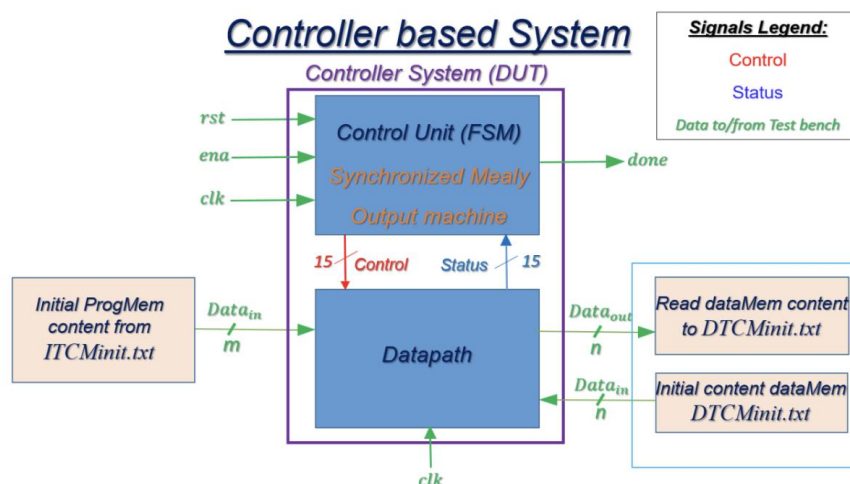


Figure 1: Overall DUT structure

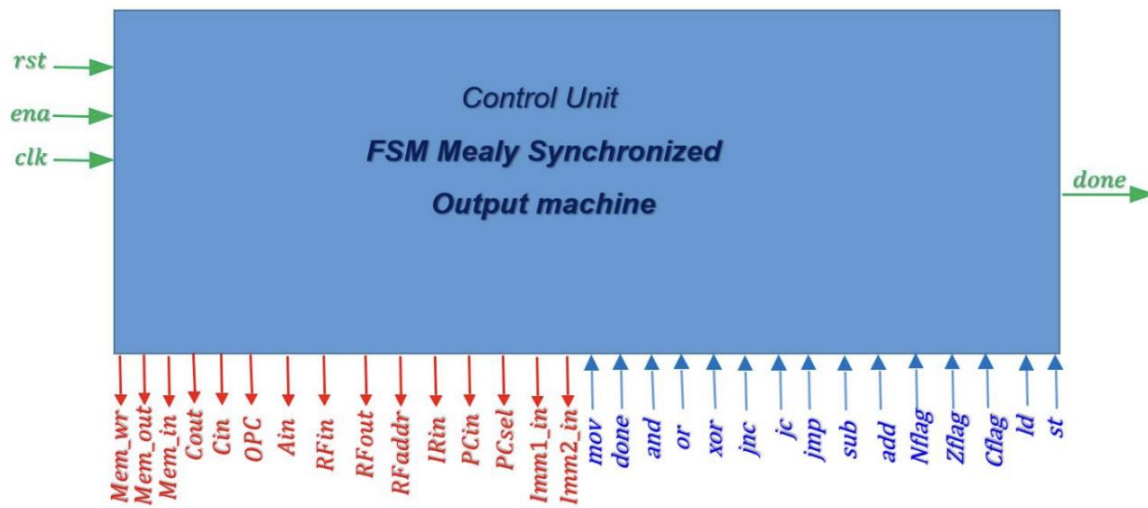
נזין את המערכת ע"י שני קבצי טקסט נתונים המכילים:

- ITCMinit.txt קובץ שיכיל את כל ה Instructions של התוכנית, אותם נכניס לתוך ה program memory.
- DTCMinit.txt קובץ שיכיל את כל המידע שישב בתאי הזיכרון לפי הסדר, אותם נכניס לתוך ה data memory.
- rst, clk, ena, done

בסיום הרצת התוכנית המערכת תוציא קובץ טקסט שמכיל את זיכרון ה DataMem של המערכת בסיום.

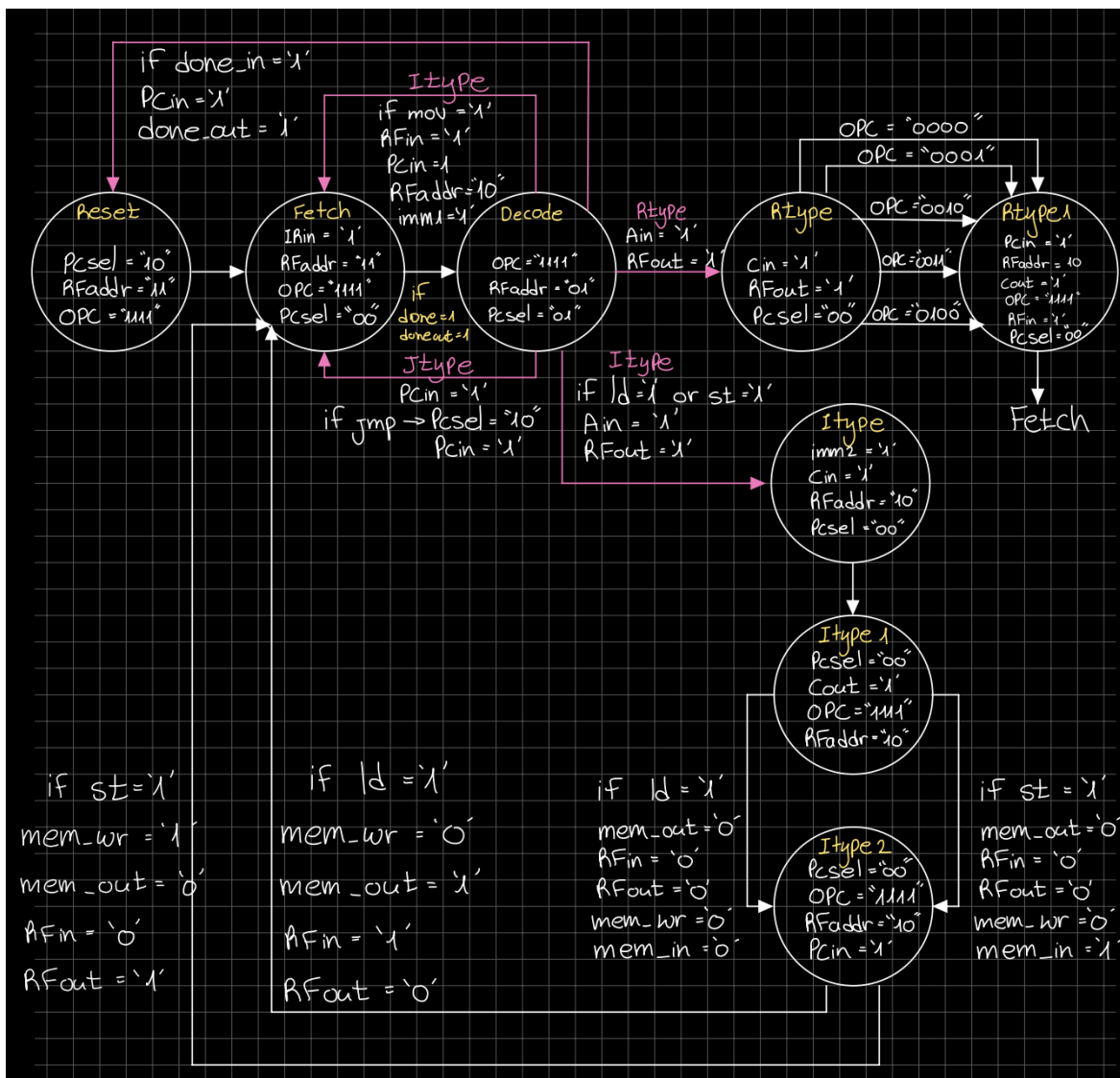
Control Unit

דיאגרמת בלוק –



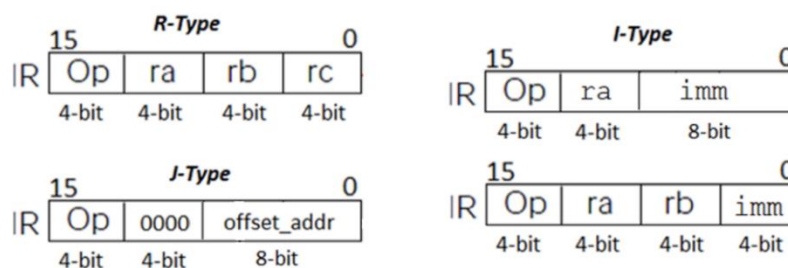
- באדום רואים קווי בקרה כפלט
- בכחול רואים קווי סטטוס כקלט
- בירור רואים את קווי בקרת test bench

תיאור המודול - מטרת המודול היא לקבל את דגל הפקודה אותה לבצע ובהינתן הדגל הזה לעלות קווי בקרה מתאימים בהתאם לFSM שמימשנו, כאשר קווי הבקרה האלו עוברים ליחידת Datapath ומכתיבים לה את אופן פעילותה.



Fetch – חישוב כתובת הפקודה הבאה לביצוע, וקריאת הפקודה הבאה לביצוע מהזיכרון.

Decode – פיענוח הפקודה ושליחתה ליחידת הבקרה. כמו כן, הבאת ערכי הרגיסטרים בהם הפקודה משתמשת בחלק מהפקודות.



דיאגרמת בלוק –



מטרת המודל לקבל מהprogram פקודה ובאמצעות decoder להוציא את דגלי הסטטוס של הפקודה והעברתה ליחידת control. בנוסף, מטרה נוספת היא בהינתן קווי בקרה לבצע את הפקודות השונות שיביאו לתוצאה הנדרשת.

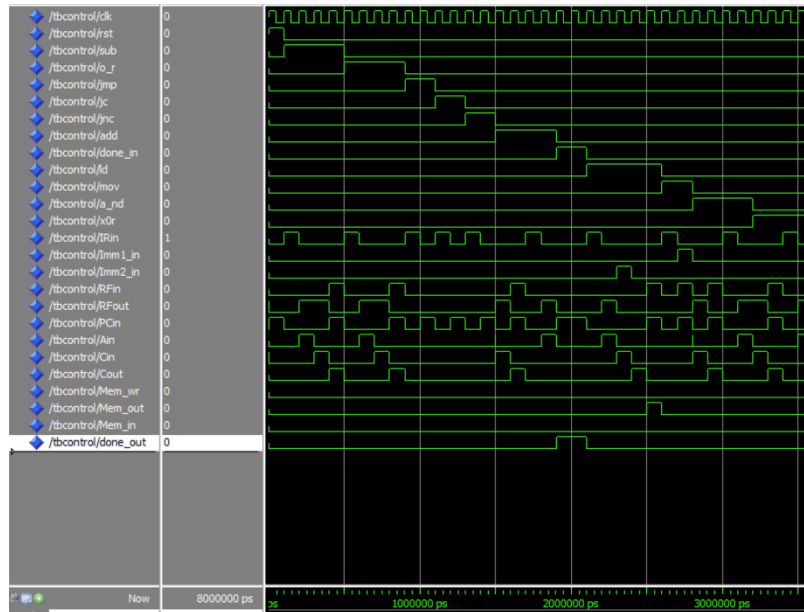
פעולות מודול זה מתוארות בדיאגרמת המצבים המצורפת מעלה, עפ קווי הבקרה השונים הנדלקים ע"י יחידת הקונטרול.

תוצאות סימולציה

סימולציה עבור ה-Control Unit

בסימולציה זו שתלנו פקודות באופן מלאכותי ובכך נוכל להבחין במספר מחזורי השעון שלוקחת כל פקודה ואילו קווי בקרה היא מדליקה בעקבות הפעלתן.

.sub → or → jmp → jc → jnc → add → done → ld → mov → and → xor



פקודת sub

נשים לב בדיאגרמת המצבים שפקודה זו לוקחת 4 מחזורי שעון כמו שרואים, קווי הבקרה שנדלקים תואמים את הנדרש לפי דיאגרמת המצבים.

פקודת jmp

נשים לב בדיאגרמת המצבים שפקודה זו לוקחת 2 מחזורי שעון כמו שרואים, קווי הבקרה שנדלקים תואמים את הנדרש לפי דיאגרמת המצבים.

פקודת ld

נשים לב בדיאגרמת המצבים שפקודה זו לוקחת 5 מחזורי שעון כמו שרואים, קווי הבקרה שנדלקים תואמים את הנדרש לפי דיאגרמת המצבים.

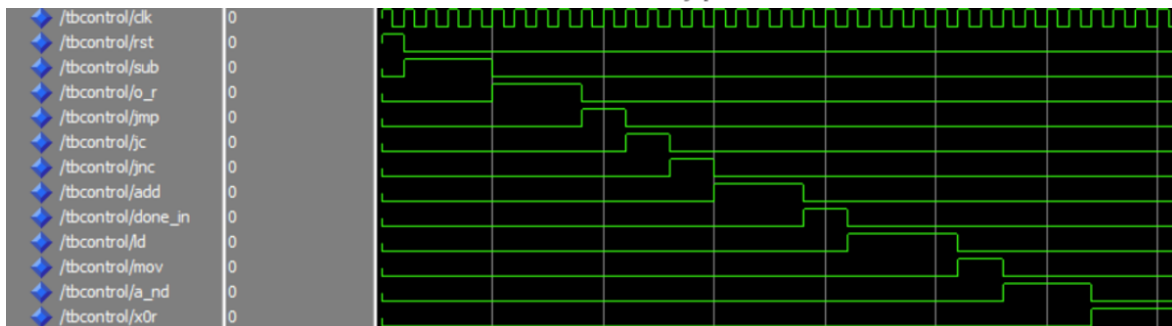
סימולציה עבור ה-Datapath Unit

בסימולציה זו הזנו את המערכת בפקודות באופן מלאכותי (פקודות זהות לפקודות שבקובץ ההרצה הנתון), ע"י העלאת הדגלים של קווי הבקרה, ע"י כך נוכל להבחין האם מיחידה זו עובדת באופן עצמאי ותקין.

ערכי הדגלים נקבעו ע"י דיאגרמת המצבים מעלה.

```
data segment:
arr dc16 20,11,2,23,14,35,6,7,48,39,10,11,12,13
res ds16 1
```

```
code segment:
ld r1,4(r0)
ld r2,5(r0)
mov r3,31
mov r4,1
mov r5,14
and r1,r1,r3
and r2,r2,r3
sub r6,r2,r1
jc 2
add r6,r4,r0
jmp 1
add r6,r0,r0
st r6,0(r5)
done
nop
jmp -2
```



נשים לב שכל הפקודות לקחו מחזורי שעות כפי שציפינו בדיאגרמת המצבים.

סימולציה עבור top

נזין את המערכת ע"י שני קבצי טקסט נתונים המכילים:

- ITCMinit.txt קובץ שיכיל את כל ה Instructions של התוכנית, אותם נכניס לתוך ה program memory.
- DTCMinit.txt קובץ שיכיל את כל המידע שישב בתאי הזיכרון לפי הסדר, אותם נכניס לתוך ה data memory.
- rst, clk, ena, done

בסיום הרצת התוכנית המערכת תוציא קובץ טקסט שמכיל את זיכרון ה DataMem של המערכת בסיום.

קבצי הטקסט המזינים את המערכת:

ITCMinit.txt	DTCMinit.txt
1 D104	1 0014
2 D205	2 000B
3 C31F	3 0002
4 C401	4 0017
5 C50E	5 000E
6 2113	6 0023
7 2223	7 0006
8 1621	8 0007
9 8002	9 0030
10 0640	10 0027
11 7001	11 000A
12 0600	12 000B
13 E650	13 000C
14 F000	14 000D
15 0000	15 0000
16 70FE	

פסודו קוד ב C:

```
int arr[14]={20,11,2,23,14,35,6,7,48,39,10,11,12,13}
int res;

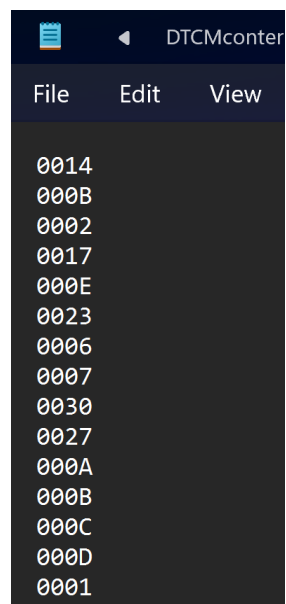
void main(){

    R[1] = arr[4] & 31;
    R[2] = arr[5] & 31;

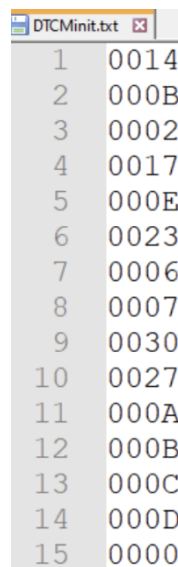
    if(R[2] >= R[1])
        res=0;
    else
        res=1;

    loop_forever;

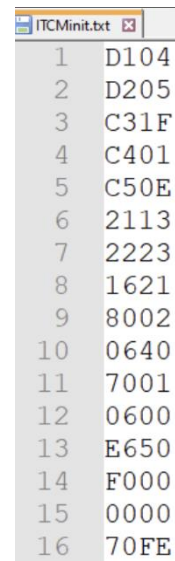
}
```



File	Edit	View
0014		
000B		
0002		
0017		
000E		
0023		
0006		
0007		
0030		
0027		
000A		
000B		
000C		
000D		
0001		



DTCMinit.txt	
1	0014
2	000B
3	0002
4	0017
5	000E
6	0023
7	0006
8	0007
9	0030
10	0027
11	000A
12	000B
13	000C
14	000D
15	0000



ITCMinit.txt	
1	D104
2	D205
3	C31F
4	C401
5	C50E
6	2113
7	2223
8	1621
9	8002
10	0640
11	7001
12	0600
13	E650
14	F000
15	0000
16	70FE