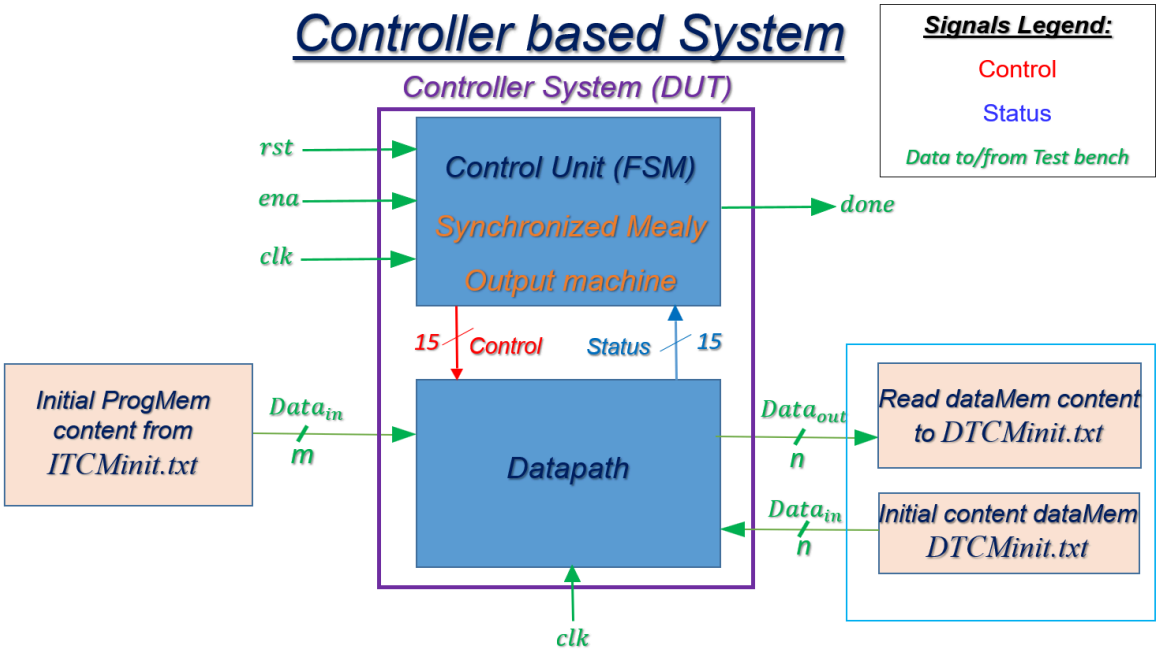
Preparation Lab3

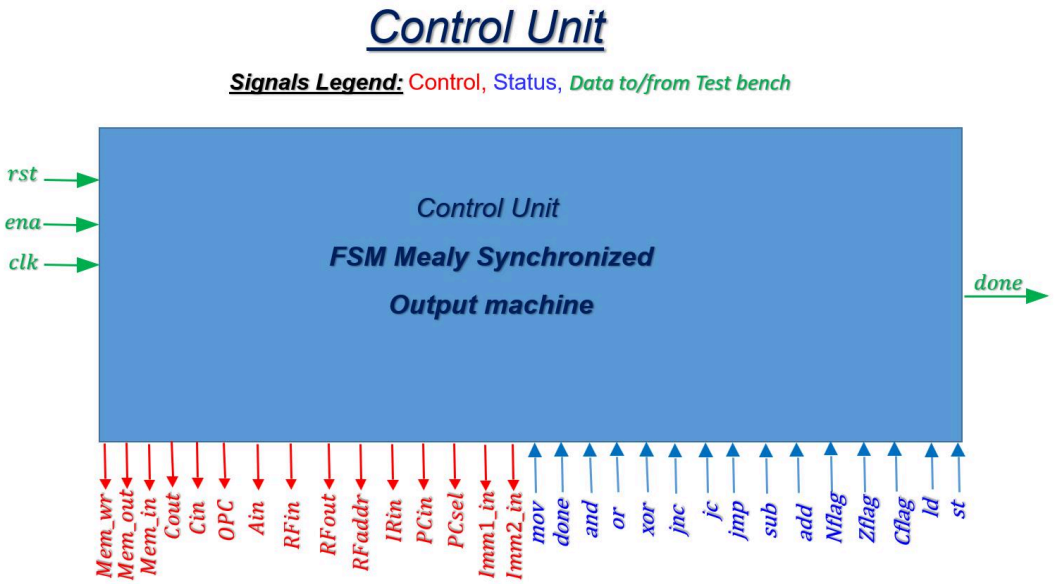
Advanced CPU architecture and hardware accelerators lab

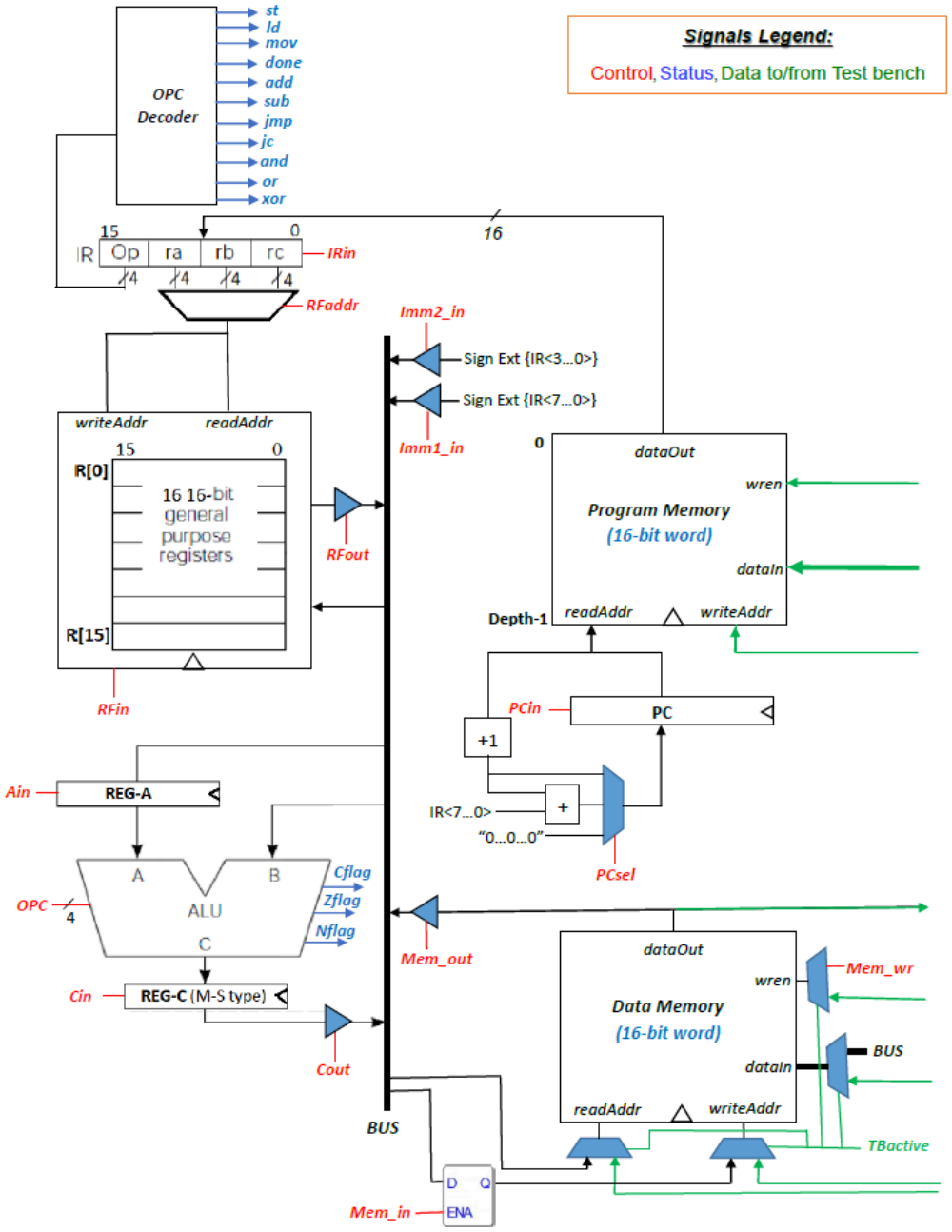
Prepared by: Yarin Oziel, Itay Kandil

הקדמה

במעבדה זו נלמד לבנות מעבד Multicycle-MIPS בשפת תיאור חומרה VHDL
ע"י הארכיטקטורה הבאה:





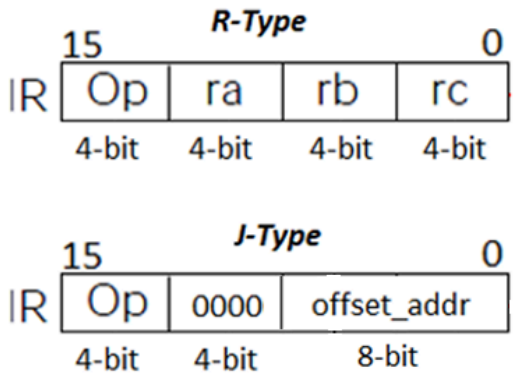


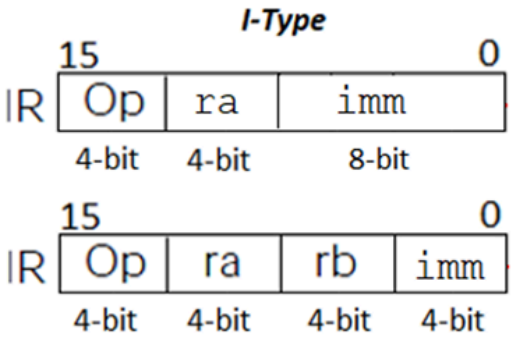
כניסות ומוצאים של המערכת:

- אות כניסה Data_in_mem: אות כניסה המזין את המצב הנוכחי של ה RAM
- אות Data_in_prog: אות כניסה המזין את התוכנית
- קו שעון clk: פעולות סינכרוניות מתבצעות בעליית השעון מ-0 ל-1
- קו בקרה rst: מאפס את כניסות המערכת בפעולה א-סינכרונית כאשר 1
- קו בקרה ena: מאפשר שימוש במודול כאשר 1

- אות מוצא Data_out_mem: אות כניסה הכותב ל RAM
- אות בקרה done: האומר למערכת שהתוכנית הסתיימה

בעמוד הבא: טבלה של ISA המסבירה את הפקודות הנתמכות במעבד ואת ה OPCODE .





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

בעמודים הבאים נראה פירוט של המודולים ביחד עם סימולציות ב MODEL-SIM
סימולציות מסוג LIST מצורפות בתיקיית DOC עקב גודל רב.

מודול Control

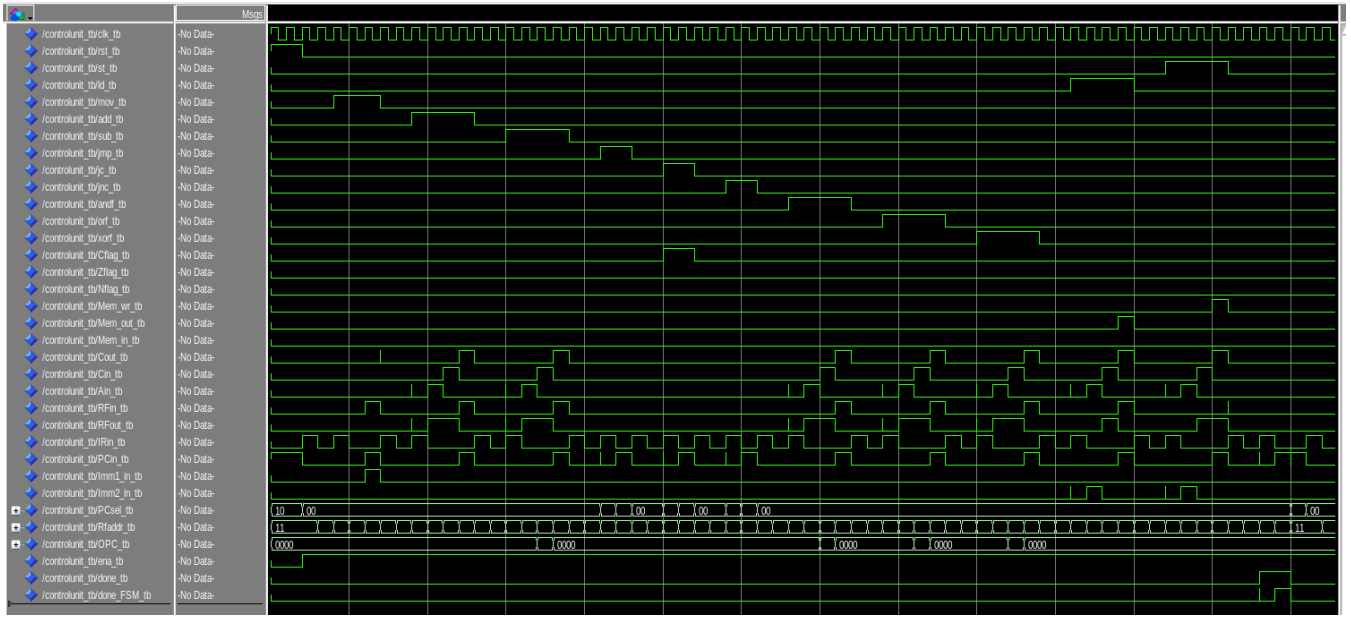
מודול זה מתפקד בתור ה"מוח" האחראי להחלפת המצבים תוך כדי שימוש במצב הנוכחי של המערכת ודגלי הסטטוס אותם הוא מקבל מיחידת Datapath.
המערכת מוגדרת בתור מערכת סינכרונית אשר מריצה תהליכים בעליית השעון.
כאשר enable מוגדר גבוה ו reset מוגדר נמוך המערכת תפעל לפי ה FSM המצורף.

סימולציה

נבדוק את תקינות המודול לפי כל המקרים האפשריים בכניסת דגלי סטטוס שונים ומצבים שונים של המערכת.
בדקנו את כל הפונקציונליות במערכת ז"א: מעבר תקין בין מצב למצב בהתאם לסוג הפעולה (RTYPE,ITYPE,JTYPE).

סימולציה מסוג WAVE:

Waveform 10us (Binary Radix):



מודול Datapath

מודול זה מתפקד בתור מערכת הגישור בן תתי המודולים השונים. תפקידה הראשוני הוא לקבל את הפקודה מה program_mem ולהפיק ממנה status signals אשר נשלחים אל ה control ומכוונים את "כיוון התנועה " על פני ה FSM .

בכל מחזור שעון , כתלות בסיגנלי הבקרה המתקבלים מן ה control, (וגם בצורה אסינכרונית עבור סיגנל ה rst) המודול פותח וסוגר באפרים שונים כדי להכווין זרימת מידע - אל ומתוך תתי המודולים כדי לבצע חישובים ופעולות התואמות את המצב הנוכחי בfsm .

לשם המחשה נציג דוגמא של פעולת ה datapath על פני מספר מחזורי שעון (ביצוע פעולת add)

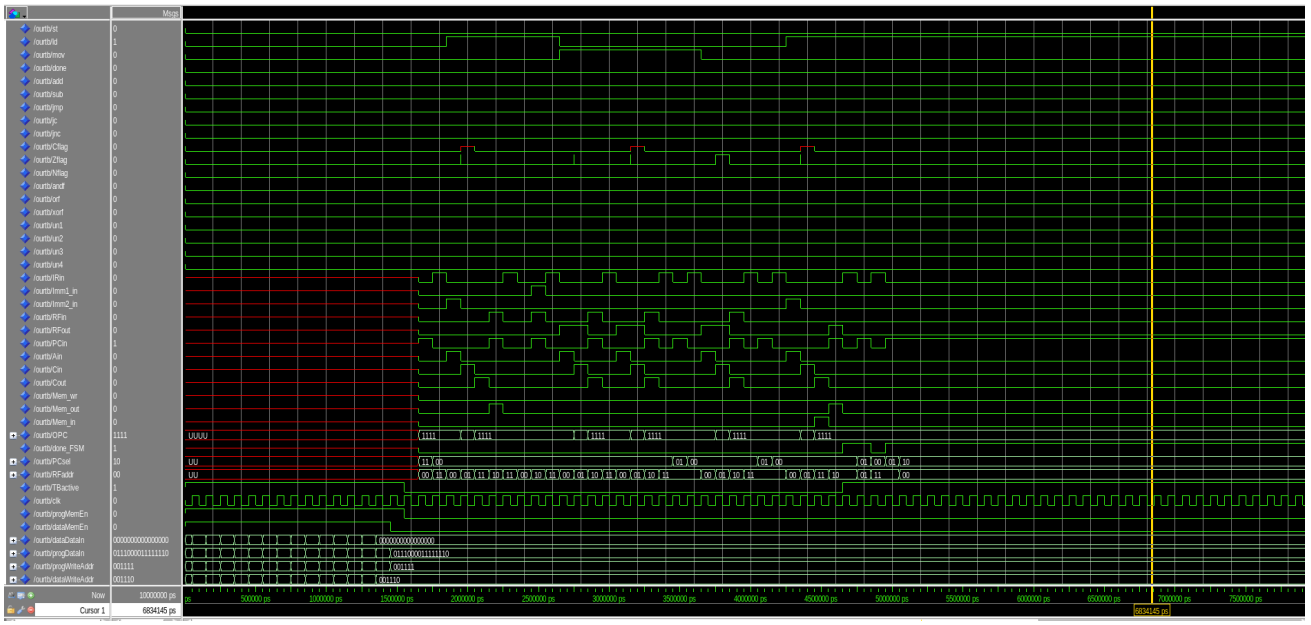
1. ה datapath מקבל את הפקודה הנוכחית מה programMem
2. ה datapath מפרש את הפקודה ומעלה את דגל ה add ל1 , מחלץ מן ה rf את תחולת rc ושומר אותה בריגסטר reg-a
3. ה datapath מחלץ מן ה rf את תחולת rb ושומר את תוצאות חיבור rc ו rb ב reg-c
4. ה datapath שומר את תוצאות החיבור בתוך ra

סימולציה

נבדוק את תקינות המודול לפי כמה מקרי קצה, הכנסנו דגלי בקרה כך שהסטטוסים ישתנו.

סימולציה מסוג WAVE:

Waveform 10us (Binary Radix):



מודול Top

מודול זה מתפקד בתור מעטפת וגישור עבור המודולים ונותנת ממשק לכניסות ומוצאי המערכת.
המערכת מוגדרת בתור מערכת סינכרונית אשר מריצה תהליכים בעליית השעון.
כאשר enable מוגדר גבוה ו reset מוגדר נמוך המערכת תקבל תוכנית כתובה ב ASSEMBLY 16BIT ורשימת זכרון המדמה את המצב הנוכחי של ה RAM, מבצעת את התוכנית וכותבת את הערך הנוכחי של הזיכרון לקובץ.

סימולציה

נבדוק את תקינות המודול לפי פלט תקין בכתיבה ל RAM.
תחילה הטסט מכניס את ה RAM והתוכנית לזכרון המערכת, לאחר מכן מריצה את הפקודות באופן סיריאלי ולבסוף כותבת את הערך הנוכחי של הזיכרון ל RAM.

סימולציה מסוג WAVE:

Waveform 10us (Binary Radix):