Lab Report

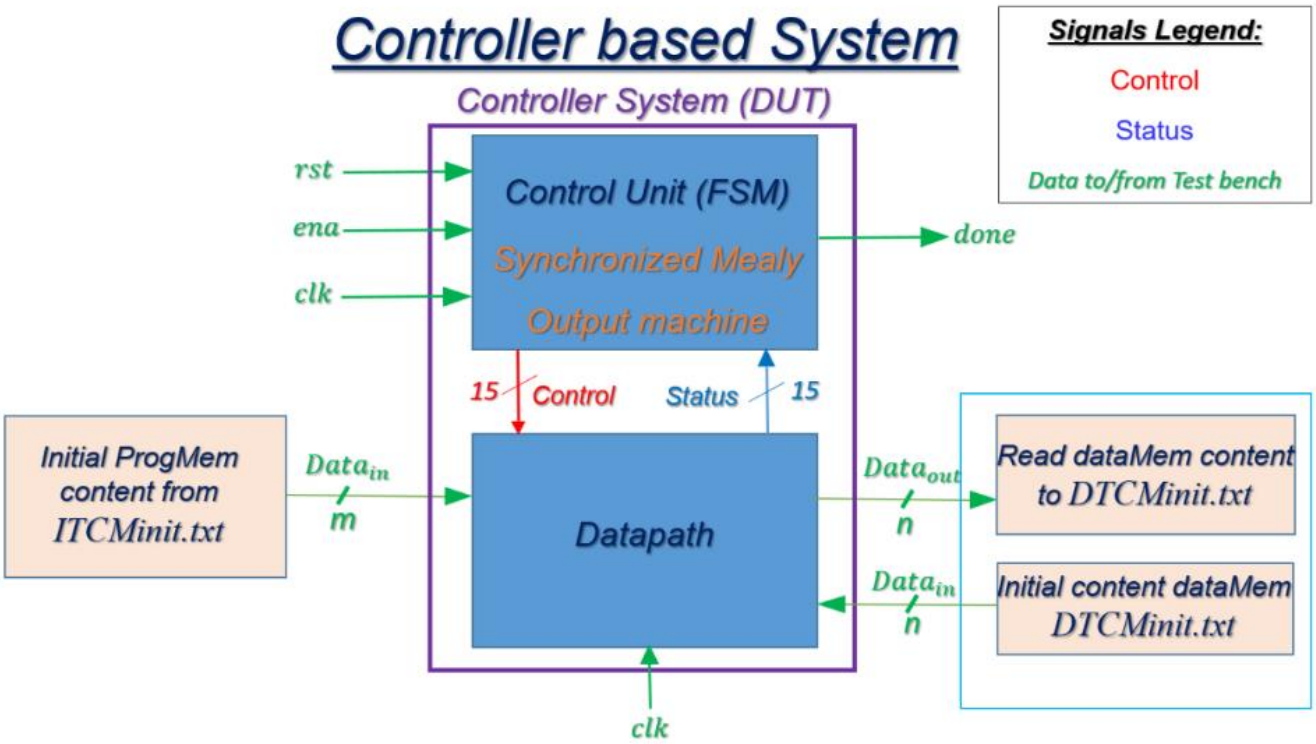
LAB 3 - DIGITAL SYSTEM DESIGN WITH VHDL
(MULTI-CYCLE CPU DESIGN)

Hagai Joseph- 207838178
Bar Kupferschmied - 318912193

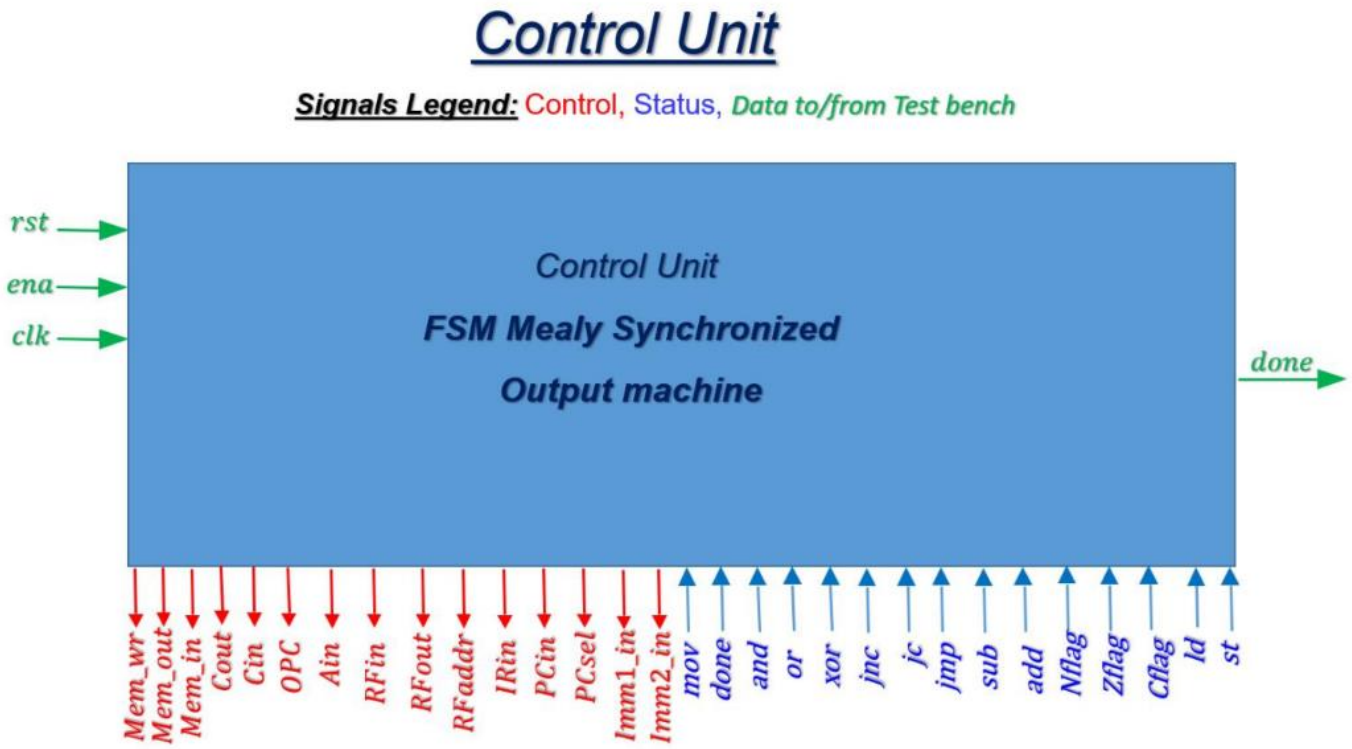
מבוא

במעבדה זו, תכננו מכונת עיבוד מבוססת בקר כמעבד רב-מחזורי על מנת להריץ קוד תוכנה נתון.

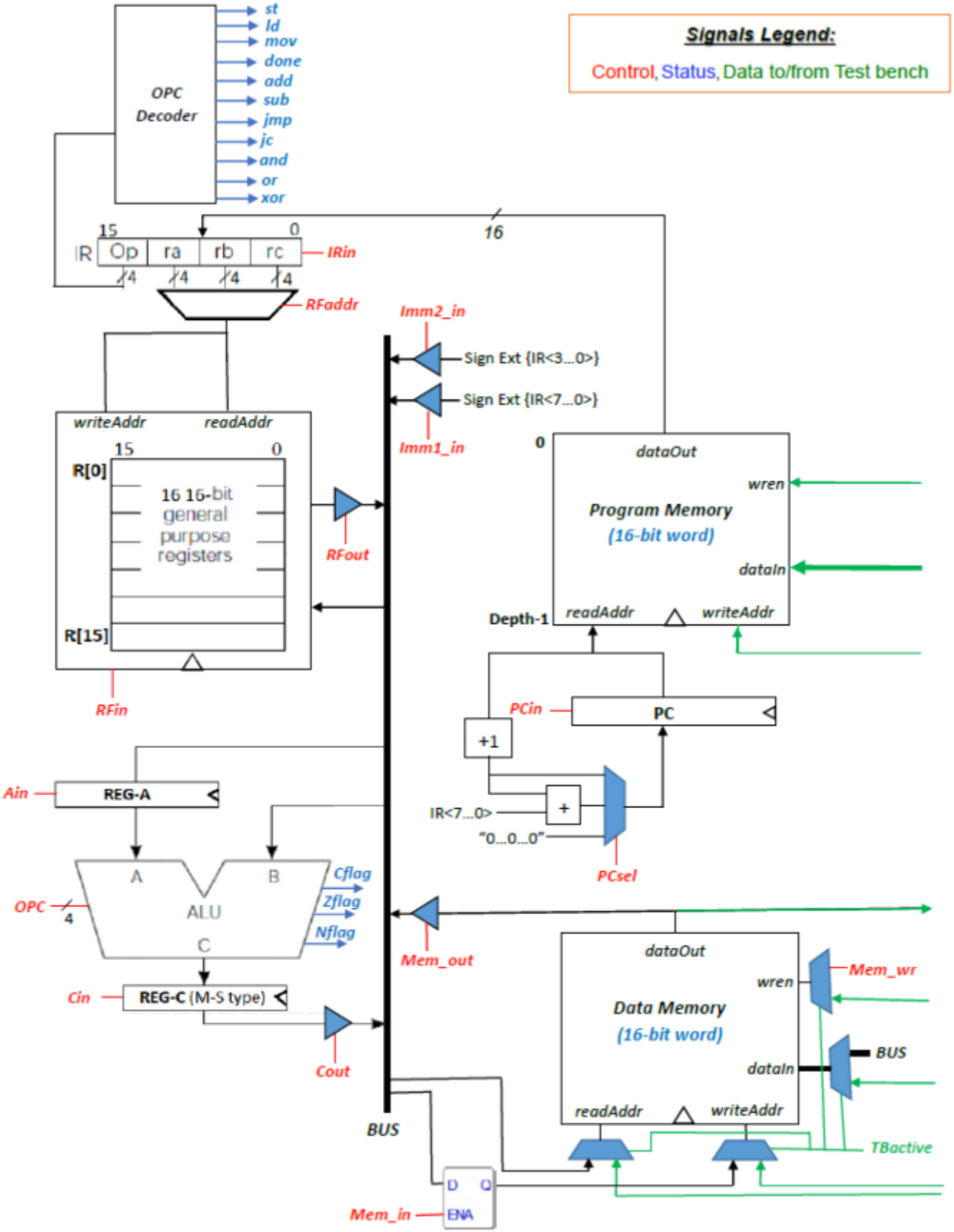
דיאגרמת המערכת:



מבנה יחידת השליטה:



מבנה הDatapath:



תיאור המערכת

המערכת מורכבת מיחידת הDatapath ויחידת הControl.

DATAPATH:

יחידה זו מורכבת ממודולים שונים שמחוברים יחד דרך הBus על מנת לממש את פעולות המערכת:

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

בהינתן אותות החיווי מהControl הDatapath מאפשר הכנסת/הוצאת ערכים מהמודולים השונים מ/אל הBus על מנת לבצע את הפעולות הנדרשות מהמערכת כולה.

מודול הProgram Memory:

זיכרון בו מאוחסנות פקודות המערכת – מודול זה ניתן לנו. בכל עליית שעון המודול מעביר את הפקודה המתאימה למודול הIR. כתובת הפקודה המתאימה ניתנת לו על ידי מודול הPC (מצביע המערכת). בנוסף ניתן לכתוב את הפקודות של התוכנית למודול על ידי חיבור חיצוני למערכת.

מודול הData Memory:

זיכרון המערכת – מודול זה ניתן לנו. בכל עליית שעון, בהינתן החיווי המתאים ניתן לקרוא/לכתוב מידע מ/אל הזיכרון. המידע והכתובת ניתן על ידי הBus. בנוסף ניתן לכתוב (ולקרוא) לזיכרון על ידי חיבור חיצוני.

מודול הPC:

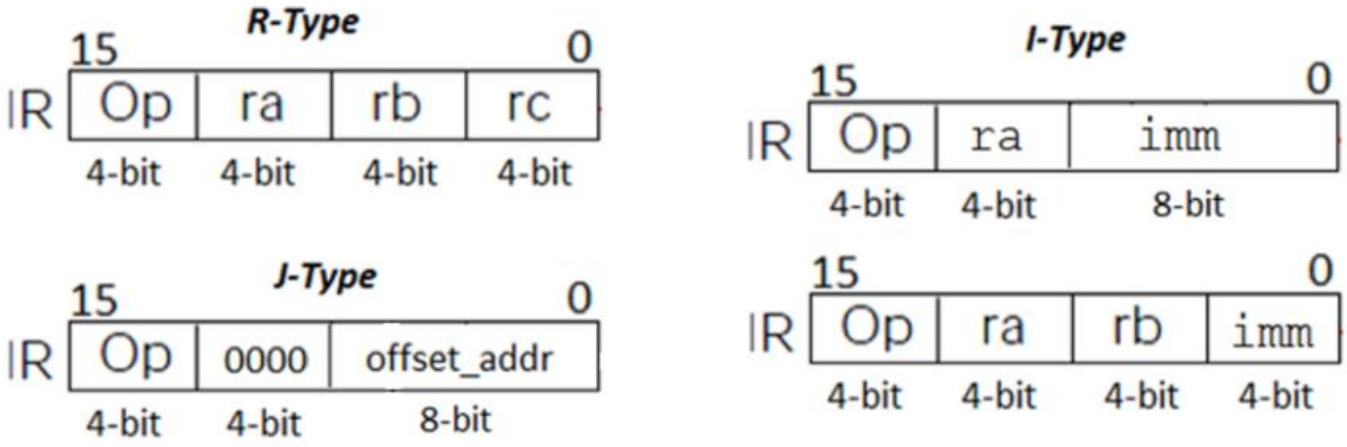
מצביע המערכת. בכל עליית שעון והינתן החיווי PC_{in} הPC מתעדכן לערך הבא שנבחר על ידי חיווי הPC_{sel}:

1. PC+1
2. PC+1+offset (שניתן על ידי הIR)
3. וקטור אפסים – איפוס המערכת

את הערך (כתובת הפקודה הבאה) הוא מעביר אל מודול הProgram Memory.

מודול הIR:

מודול זה מקבל את השורה הבאה בתוכנית ממודול הPC ומעביר את ביטי הפקודה אל מודול הDecoder (מפענח המערכת). בנוסף הIR מעביר את הקבועים/כתובות הרגיסטרים/הoffset אל מחזיק הקבועים/מודול הRegister File (בהתאמה) לפי התבנית:



מודול הDecoder:

מודול זה מקבל את ביטי הפקודה ממודול הIR, ושולח ליחידת הControl את הפקודה שהמערכת צריכה לבצע. (יחידת הControl מוציאה את ביטי בחיווי המתאימים לביצוע הפעולה).

מודול ה(Register File (RF:

מודול זה מכיל את הרגיסטרים של יחידת העיבוד - מודול זה ניתן לנו. בהינתן החיווי RF_{out} וכתובת רגיסטר לקריאה המודול מוציא את המידע המתאים לBus. בהינתן כתובת רגיסטר לכתיבה והחיווי RF_{in} המודול כותב את המידע מהBus לכתובת המתאימה.

מודול הALU:

מודול זה מבצע פעולות אריתמטיות על שתי כניסות A וB ואת התוצאה הוא מעביר לC. המודול מבצע את הפעולה לפי חיווי OPC ובנוסף מעביר דגלי Zflag,Nflag,Cflag אל יחידת הControl. כניסות A וB מתקבלות דרך הBus כאשר בA יש רגיסטר (לשמירת הערך עד לביצוע הפעולה) וB מחובר ישירות אל הBus.

מודול הImmediate:

מודול זה מחזיק הרחבת סימן של קבועים שניתנים לו מהIR ובהינתן החיווי המתאים מיחידת הControl מכניס אותם לBus.

תוצאות סימולציה:

לסימולציית TB הDatapath קראנו מתוך קובץ ערכים לזיכרון המערכת (מידע ותוכנית) וביצענו את הפעולות כך שהכנסנו ידנית את אותות יחידת הControl. ביצענו את הקוד הבא:

```
-------------------------------------------------------------------------------
-- This test bench check the folowing Instruction:
-------------------------------------------------------------------------------
--     1. mov   R1,16       = 0xC110    (R1=16)
--     2. mov   R2,17       = 0xC211    (R2=17)
--     3. ld    R3,0(R0)    = 0xD300    (R3=1)
--     4. ld    R4,0(R1)    = 0xD410    (R4=0xF0F0)
--     5. ld    R5,1(R1)    = 0xD511    (R5=0xFFFF)
--     6. st    R1,0(R1)    = 0xE110    (store in addres 16)
--     7. st    R2,0(R2)    = 0xE220    (store in addres 17)
--     8. add   R1,R3,R1    = 0x0131    (R1=17)
--     9. sub   R1,R2,R1    = 0x1121    (R1=0)
--    10. st    R1,1(R2)    = 0xE121    (store in addres 18)
--    11. and   R6,R5,R4    = 0x2654    (expecting: 0xF0F0)
--    12. or    R7,R5,R4    = 0x3754    (expecting: 0xFFFF)
--    13. xor   R8,R5,R4    = 0x4854    (expecting: 0x0F0F)
--    14. st    R6,2(R2)    = 0xE622    (store in addres 19)
--    15. st    R7,3(R2)    = 0xE723    (store in addres 20)
--    16. st    R8,4(R2)    = 0xE824    (store in addres 21)
--    17. done              = 0xF000    (exiting)
-------------------------------------------------------------------------------
```

תוכן קובץ הזיכרון:

0001
0001
0002
0003
0004
0005
0006
0007
0008
0009
000A
000B
000C
000D
000E
000F
F0F0
FFFF

תוכן קובץ הפקודות:

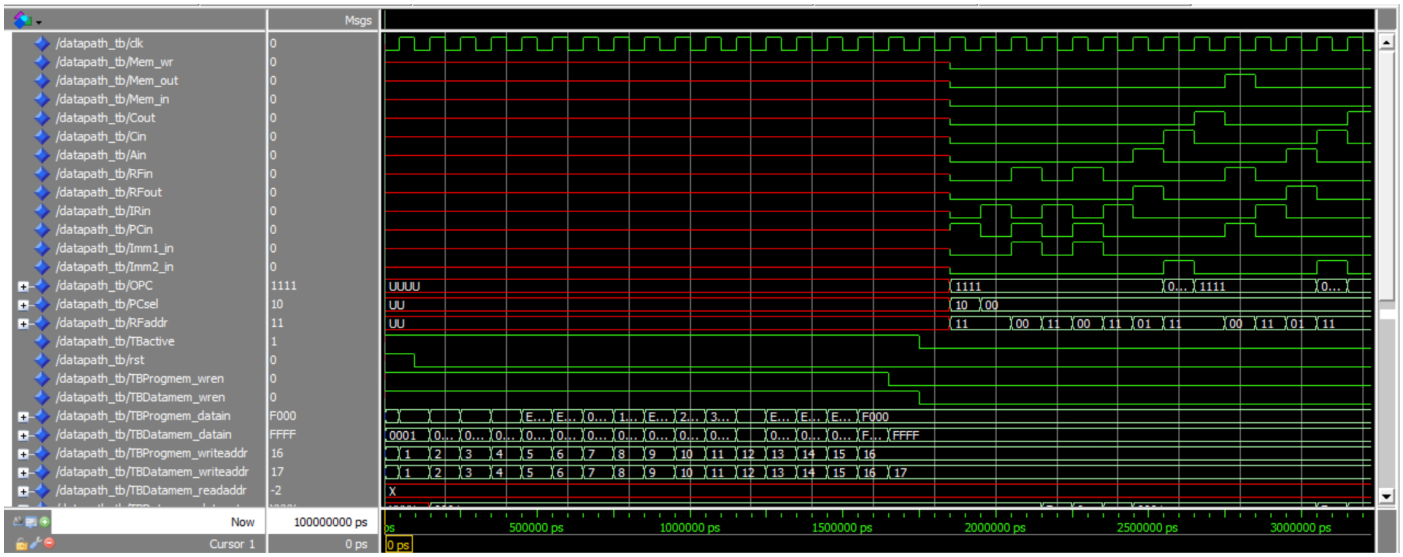
C110
C211
D300
D410
D511
E110
E220
0131
1121
E121
2654
3754
4854
E622
E723
E824
F000

תוצאות הזיכרון לאחר הרצת הסימולציה:

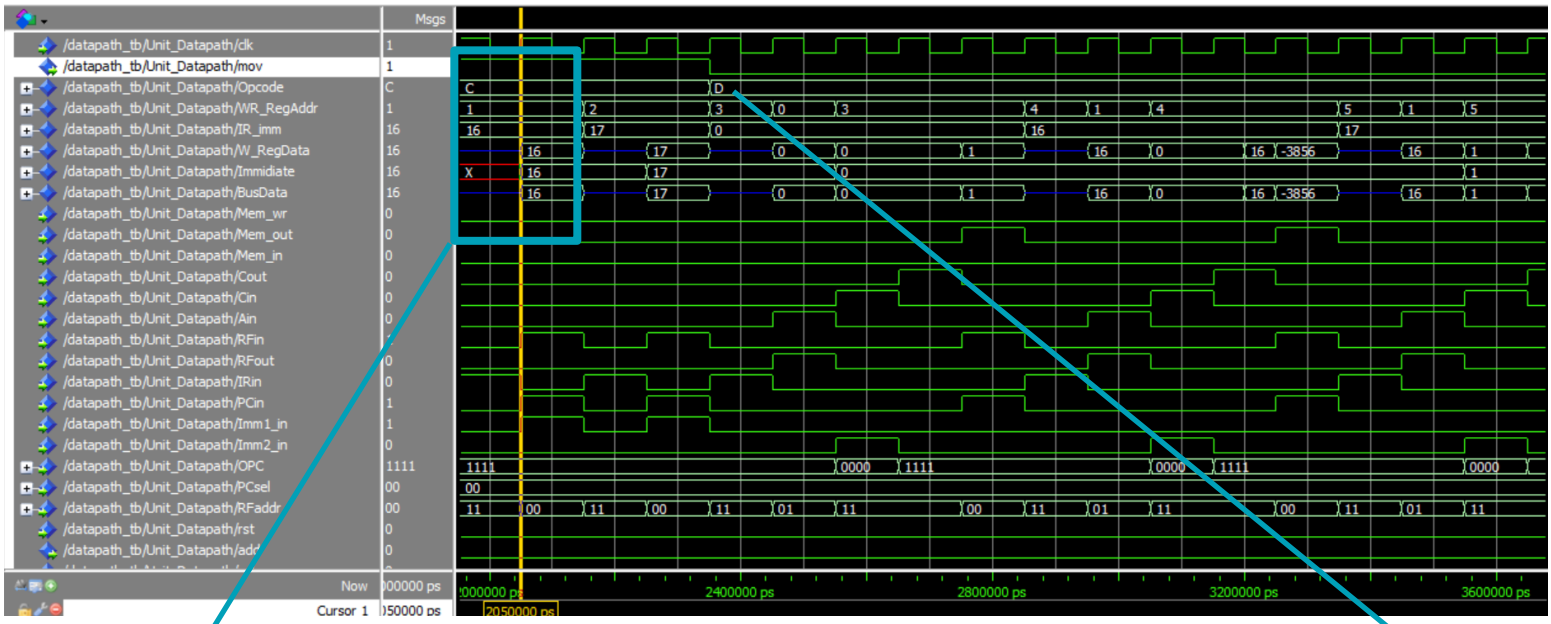
0001
0001
0002
0003
0004
0005
0006
0007
0008
0009
000A
000B
000C
000D
000E
000F
0010
0011
0000
F0F0
FFFF
0F0F

מידע חדש שנשמר

ניתן לראות כי לאחר הרצת הסימולציה נשמרו לנו ערכים חדשים (כפי שהיינו מצפים מהם להיות) במקומות 16-21) בזיכרון כפי שרצינו.



טעינת המידע לזיכרון
מן הקבצים



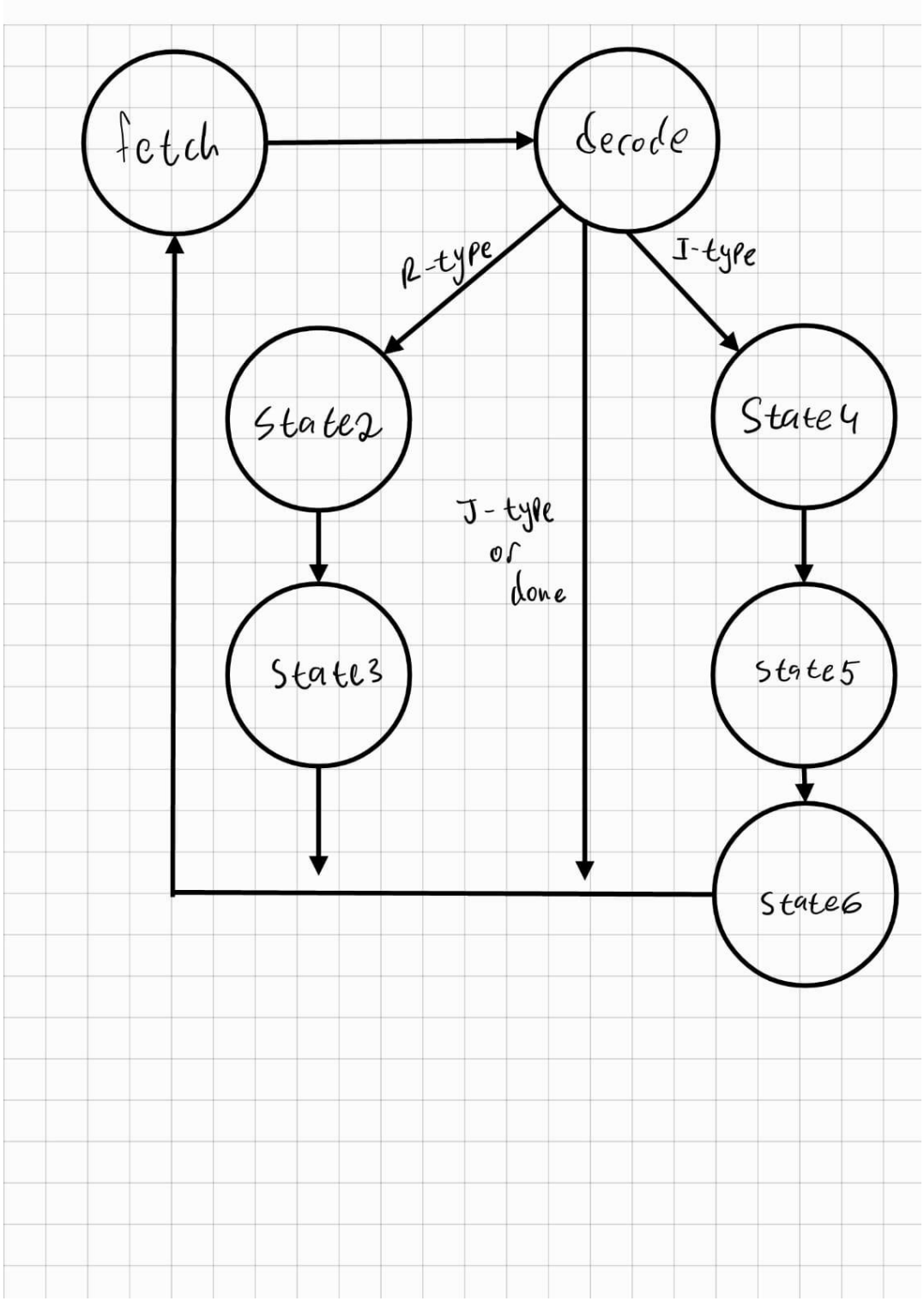
ניתן לראות במסגרת כי פעולת הmov מתרגמת נכון בDecoder ובנוסף הערך המתאים נטען לImmediate ולBus. הרגיסטר הנכון לאחסון הקבוע נבחר נכון ובעצם הפעולה מבוצעת כמו שצריך.

ניתן לראות כי הפעולת הload מתורגמת נכון בDecoder

CONTROL:

המודול אחראי על מכונת המצבים אשר שומרת על ביצוע נכון של פעולות המעבד. הוא מבצע זאת על-ידי שימוש באותות בקרה אשר מאפשרים כניסה של אותות לBus או לכניסה/יציאה של רגיסטרים. בהתאם לכניסות והמצב הנוכחי, המכונת מצבים מוציאה אותות בקרה לרכיבים ומעדכנת את המצב הבא.

סכמה אשר מתארת את מכונת המצבים, ופירוט האותות אשר מוצאים בכל מצב:



fetch

IR in = 1
opcode, RFaddr = unaffected
Other = 0
Next = decode

decode

R-type:

A in = 1
RFout = 1
RFaddr = 01
opc = unaffected
others = 0
Next: S2

Done:

PC in = 1
PCSel = 01
fsmDone = 1
others = 0
Next: fetch

Move:

Imm1_in = 1
RF in = 1
Pc in = 1
others = 0
Next = fetch

Load or Store:

Imm2_in = 1
A in = 1
other = 0
Next: S4

J-type

PC in = 1
others = 0
if the jump condition is satisfied: PCSel = 01
otherwise PCSel = 00

Next: fetch

R-type continuation

S2: R-type execution

Cin = 1
RFout = 1
RFaddr = 10
OPC = According to operation
Others = 0
Next: S3

S3: R-type finish

Cout = 1
RFin = 1
PCin = 1
Others = 0
Next: fetch

I-type continuation

S4: I-type part 1

Cin = 1
RFout = 1
RFaddr = 01
OPC = 0000
Other = 0
Next = S5

S5: I-type part 2

Cout = 1
if Store: Mem_in = 1
Others = 0
Next: S6

S6: I-type part 3

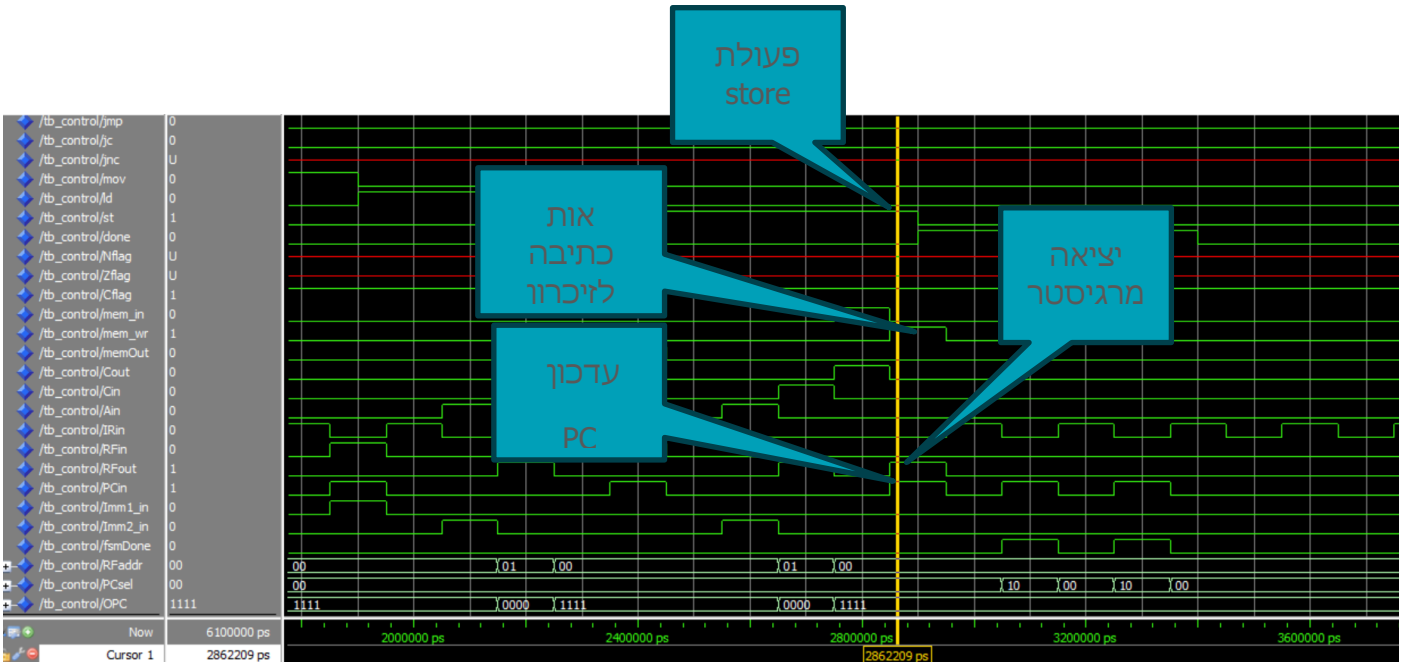
PCin = 1
if load: Mem_out = 1, RFin = 1
if Store: Mem_WR = 1, RFout = 1
Others = 0
Next = fetch

Reset:

PCin = 1
PCsel = 10
Others = 0
Next = fetch

תוצאות הסימולציה:

עבור הדוגמא הבאה, במחזור האחרון של פעולת הכתיבה לזיכרון, נשים לב כי דלוקים האותות של הפעולה, הכתביה לזיכרון, היציאה מהרגיסטר (שם המידע) ועדכון הPC, מכיוון שנמשיך לאחר המחזור הזה חזרה לfetch.



תוצאות סימולציה מהTop:

את שתי היחידות – הDatapath והControl מחברים ביחידת הTop. בהרצת הקוד הבא:

```
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)   -- R1<=14
ld  r2,5(r0)   -- R2<=35
mov r3,31      -- R3<=31
mov r4,1       -- R4<=1
mov r5,14      -- R5<=14
and r1,r1,r3   -- R1<=14
and r2,r2,r3   -- R2<=3
sub r6,r2,r1   -- R6<=-11
jc  2          -- carry is 0 => no jump
add r6,r4,r0   -- R6<=1
jmp 1          -- jump to store
add r6,r0,r0
st  r6,0(r5)   -- storing R6=1 in arr[14] (after 13 in memory)
done
nop
jmp -2
```

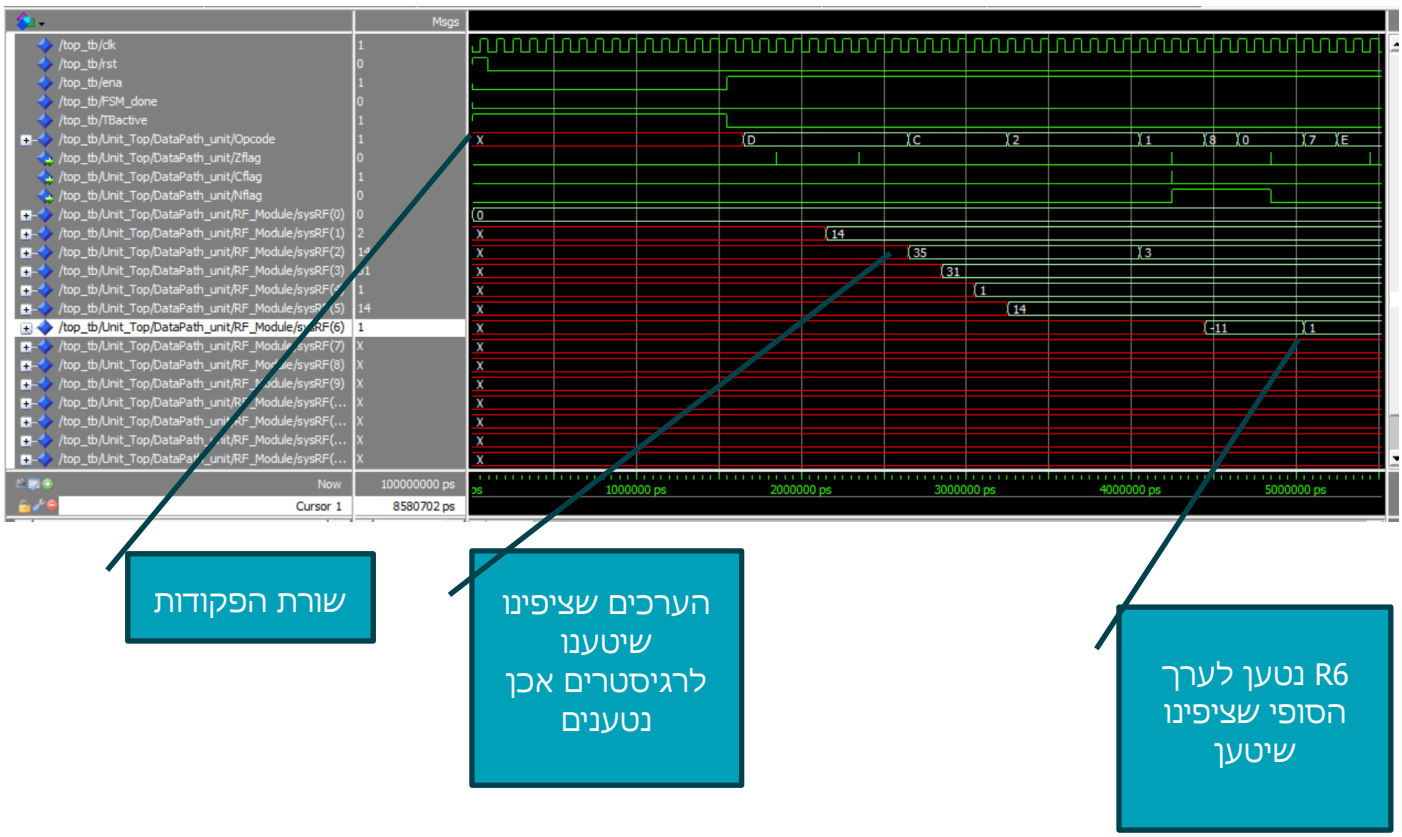
המתורגם לתוכנית:

```
D104
D205
C31F
C401
C50E
2113
2223
1621
8002
0640
7001
0600
E650
F000
0000
70FE
```

וטעינת הזיכרון הבא למערכת:

```
0014
000B
0002
0017
000E
0023
0006
0007
0030
0027
000A
000B
000C
000D
0000
```

נראה שבאמת נקבל את המצופה:



וניתן לראות זאת גם בקובץ שנשלוף לתוכו את הזיכרון: