מעבדה 3

יואב אשד 305384869

גלעד בינו 302252101

מבוא

מעבד – יחידת עיבוד מרכזית, הינו רכיב חומרה במערכת מחשוב הקורא ומבצע פקודות המוזנות אליו מזיכרון המחשב.

הפקודות אותן מקבל המעבד מהזיכרון יכולות נעות בן קריאה וכתיבה לזיכרון, ביצוע פעולות לוגיות . הפקודות מורכבות מתשעה של ביטים של מידע , כאשר שלושת הביטים הראשונים מייצגים את הפקודה, שלושת הביטים האמצעיים מייצגים רגיסטר אחד ושלושת הביטים האחרונים מייצגים רגיסטר שני.

$command\ bit\ structre$

$Instruction$	$Reg\ X$	$Reg\ Y$

פקודות המעבד:

המעבד שלנו הינו מסוג multi cycle (תומך בכמה בפקודות במקביל בכל זמן מחזור) . המעבד תומך בפקודות הבאות : $mv, mvi\ , add, sub, addi, subi$ נפרט על כל אחת מהן:

MV :

פקודה המעתיקה מידע מכניסת המידע לרגיסטר, אשר מתבצעת במחזור שעון אחד.

- ברגע T_0 ,ביט IR_in דלוק , ולכן המעבד קורא את הפקודה הנכנסת מכניסת הDIN ושומר אותה ברגיסטר IR עד להתקדמות למצב הבא:
- ברגע T_1 , המעבד קורא את הפקודה מרגיסטר IR , ובהתאם , מדליק את הביט אל רגיסטר היעד, על מנת שיוכל לכתוב אליו, ולאחר מכן מעביר את המידע ליציאת ה$BUSWIRES$, ומשם המידע יועבר לרגיסטר היעד וישמר שם.
- לאחר מכן המעבד ידליק את ביט $Done$ לטובת סימון סיום הפקודה ומעבר לפקודה הבאה.

MVI:

פקודה המעתיקה מידע מרגיסטר לרגיסטר, אשר מתבצעת במחזור שעון אחד.

- ברגע T_0 ,ביט IR_in דלוק , ולכן המעבד קורא את הפקודה הנכנסת מכניסת הDIN ושומר אותה ברגיסטר IR עד להתקדמות למצב הבא:
- ברגע T_1 , המעבד קורא את הפקודה מרגיסטר IR , ובהתאם , מדליק את הביט אל רגיסטר היעד, על מנת שיוכל לכתוב אליו, ולאחר מכן מעביר את המידע ליציאת ה$BUSWIRES$, ומשם המידע יועבר לרגיסטר היעד וישמר שם.
- לאחר מכן המעבד ידליק את ביט $Done$ לטובת סימון סיום הפקודה ומעבר לפקודה הבאה.

add:

פקודה המחברת בין שני ערכי רגיסטרים, ושומרת את התוצאה ברגיסטר היעד (האמצעי בפקודה) , אשר מתבצעת בארבעה מחזורי שעון.

- ברגע T_0 ,ביט IR_in דלוק ,ולכן המעבד קורא את הפקודה הנכנסת מכניסת הDIN ושומר אותה ברגיסטר IR עד להתקדמות למצב הבא:
- ברגע T_1 , קוראים מידע מרגיסטר X , ומדליקים את הביט a_in כך שהמידע שיועבר ל buswires יישמר ברגיסטר A לצורך פעולת החישוב.
- ברגע T_2 קוראים מידע מרגיסטר Y , מכבים את הביט a_in כך שהמידע שנמצא ברגיסטר A לא ישתנה, ומדליקים את הביט של g_in על מנת שתוצאת החישוב של הALU תישמר ברגיסטר G.
- ברגע T_3 , ביט g_in נכבה, תוצאת החישוב השמורה ברגיסטר g מגיע אל ה $buswires$,באמצעות ניתוב שמתבצע על ידי הmux , נשמרת ברגיסטר המטרה. לאחר מכן נדלק ביט ה$done$ לציון סיום הפעולה.

sub :

פקודה המחברת בין שני ערכי רגיסטרים, ושומרת את התוצאה ברגיסטר היעד (האמצעי בפקודה) , אשר מתבצעת בארבעה מחזורי שעון.

- ברגע T_0 ,ביט IR_in דלוק , ,ולכן המעבד קורא את הפקודה הנכנסת מכניסת הDIN ושומר אותה ברגיסטר IR עד להתקדמות למצב הבא:
- ברגע T_1 , קוראים מידע מרגיסטר X , ומדליקים את הביט a_in כך שהמידע שיועבר ל $buswires$ יישמר ברגיסטר A לצורך פעולת החישוב.
- ברגע T_2 קוראים מידע מרגיסטר Y , מכבים את הביט a_in כך שהמידע שנמצא ברגיסטר A לא ישתנה, ומדליקים את הביט של g_inעל מנת שתוצאת החישוב של הALU תישמר ברגיסטר G.
- ברגע T_3 , ביט g_in נכבה, תוצאת החישוב השמורה ברגיסטר g מגיע אל ה $buswires$,באמצעות ניתוב שמתבצע על ידי הmux , נשמרת ברגיסטר המטרה. לאחר מכן נדלק ביט ה$done$ לציון סיום הפעולה.

addi

פקודה המחברת בין שני ערכי רגיסטרים, ושומרת את התוצאה ברגיסטר היעד (האמצעי בפקודה) , אשר מתבצעת בארבעה מחזורי שעון.

- ברגע T_0 ,ביט IR_in דלוק , ,ולכן המעבד קורא את הפקודה הנכנסת מכניסת הDIN ושומר אותה ברגיסטר IR עד להתקדמות למצב הבא:
- ברגע T_1 , קוראים מידע מרגיסטר X , ומדליקים את הביט a_in כך שהמידע שיועבר ל$buswires$ יישמר ברגיסטר A לצורך פעולת החישוב.
- ברגע T_2 קוראים מידע מ- DIN, מכבים את הביט a_in כך שהמידע שנמצא ברגיסטר A לא ישתנה, ומדליקים את הביט של g_inעל מנת שתוצאת החישוב של הALU תישמר ברגיסטר G.
- ברגע T_3 , ביט g_in נכבה, תוצאת החישוב השמורה ברגיסטר g מגיע אל ה $buswires$,באמצעות ניתוב שמתבצע על ידי הmux , נשמרת ברגיסטר המטרה. לאחר מכן נדלק ביט ה$done$ לציון סיום הפעולה.

subi

- פקודה המחברת בין שני ערכי רגיסטרים, ושומרת את התוצאה ברגיסטר היעד (האמצעי בפקודה) , אשר מתבצעת בארבעה מחזורי שעון.
- ברגע T_0 ,ביט IR_in דלוק , ,ולכן המעבד קורא את הפקודה הנכנסת מכניסת הDIN ושומר אותה ברגיסטר IR עד להתקדמות למצב הבא:
- ברגע T_1 , קוראים מידע מרגיסטר X , ומדליקים את הביט a_in כך שהמידע שיועבר ל $buswires$ יישמר ברגיסטר A לצורך פעולת החישוב.
- ברגע T_2 קוראים מידע מ- DIN , מכבים את הביט a_in כך שהמידע שנמצא ברגיסטר A לא ישתנה, ומדליקים את הביט של g_in על מנת שתוצאת החישוב של הALU תישמר ברגיסטר G.
- ברגע T_3 , ביט g_in נכבה, תוצאת החישוב השמורה ברגיסטר g מגיע אל ה $buswires$,באמצעות ניתוב שמתבצע על ידי הmux , נשמרת ברגיסטר המטרה. לאחר מכן נדלק ביט ה$done$ לציון סיום הפעולה.

$Operation$	$Function\ Preformed$
$mv\ R_xR_y$	$R_x \leftarrow [R_y]$
$mvi\ R_x, \#D$	$R_x \leftarrow D$
$add\ R_x, R_y$	$R_x \leftarrow [R_x] + [R_y]$
$sub\ R_x, R_y$	$R_x \leftarrow [R_x] - [R_y]$
$addi\ R_x, \#D$	$R_x \leftarrow [R_x] + D$
$subi\ R_x, \#D$	$R_x \leftarrow [R_x] - D$

אופן פעולת המצבים:

מכונת המצבים הממומשת פועלת באופן הבא:

בכל שלב ושלב כל האותות מוחזקים בתור ברירת מחדל ב0 , כלומר כבויים (ביטים) או ריקים (רגיסטרים).

כאשר ביט RUN דולק, המכונה בודקת את התנאים ועוברת בין המצבים, אחרת היא נשארת באותו מצב שבו הייתה.

כאשר ביט $Done$ דולק, המכונה מעדכנת את היציאה הרלוונטית וחוזרת למצב T_0.

המצבים השונים מתארים זמני מחזור של המעבד, כאשר כל זמן מחזור מיוצג על ידי שני ביטים.

במצב 00 , המכונה נמצאת במחזור T_0 , השלב ההתחלתי בו המכונה קוראת מידע מIR.

במצב 01 , המכונה נמצאת במחזור T_1 , במידה והפעולה הנתונה היא פעולת mv או mvi, ביט ה$done$ נדלק והמכונה חוזרת לT_0. אם מדובר בפקודה אחרת , המכונה ממשיכה לשלב T_2.

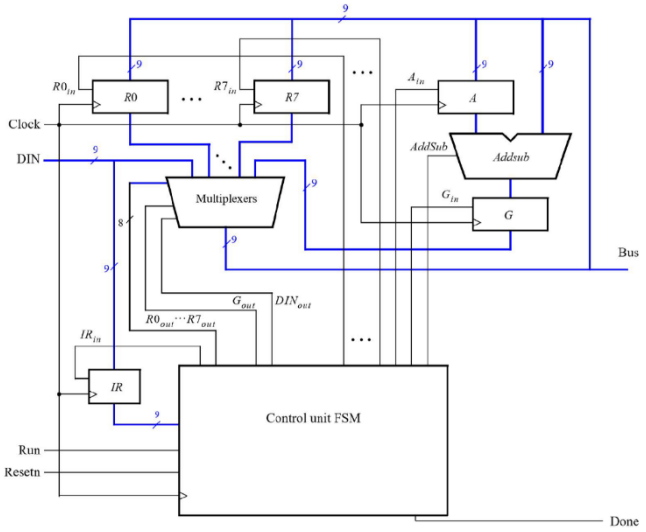
במצב 10 המכונה נמצאת במחזור T_2, כלומר מתבצעת פעולה שמערבת חישוב באמצעות ה ALU. שומרים ערך לרגיסטר A לצורך חישוב בשלב הבא.

במצב 11 המכונה נמצאת במחזור T_3, המשך פעולת החישוב, שמירת תוצאת החישוב ברגיסטר התוצאה (G) והקפצת ביט $Done$, לאחר מכן המכונה חוזרת למצב 00 , כלומר חזרה להתחלה של מחזור חדש.

טבלה המייצגת את הביטים והרגיסטרים הפעילים עבור כל מחזור:

	T_0	T_1	T_2	T_3
MV	IR_in	$RY\_out, RX\_in\ \ Done$		
MVI	IR_in	$DIN\_out, RX\_in\ \ Done$		
ADD	IR_in	$RXout\ , Ain$	$RYout\ , Gin$	$Gout\ RXin$ $DONE$
SUB	IR_in	$RXout\ , Ain$	$RYout\ , Gin$	$Gout\ RXin$ $DONE$
$ADDI$	IR_in	$RXout, Ain$	$DINout, Gin$	$Gout\ RXin$ $DONE$
$SUBI$	IR_in	$RXout, Ain$	$DINout, Gin$	$Gout\ RXin$ $DONE$

סכמת המערכת.



פירוט כניסות ויציאות:

- CLK -אות שעון המתזמן את המערכת.
- DIN – אות בן תשעה ביטים המייצג את הפקודה המגיעה למעבד.
- RUN- אות המורכב מביט אחד, המאפשר למעבד להתחיל לעבוד ולעבור בין מצב למצב.
- $RESETN$ - אות המאפס את המצב הנוכחי למצב ההתחלתי.
- $DONE$ – אות המורכב מביט אחד המסמן את סיום פעולת המעבד הנוכחית.
- $BUSWIRES$ - רגיסטר יציאה בין תשע ביטים המעביר מידע אל הרגיסטרים ואל יחידת החישוב לטובת המשך פעולת המעבד.

חלוקת המערכת למודולים:

מודול הטופ ומכונת המצבים של הפרויקט - $proc$:

משמש בתור הטופ של כל המעבד וכן בתור מכונת המצבים המבצעת את פעולות המעבד.

כניסות: $DIN, Resetn, Clock, Run,,$

יציאות: $Done, R0, R1, R2, R3, R4, R5, R6, R7, RA, RG, alu_{out}, Tstep\_Q, BusWires$

מודול הmux_module :

משמש בתור בורר יציאות וכניסות למעגל , הmux בוחר כניסה אחת לפי האותות שהוא מקבל מproc ומעביר את המידע ממנה אל הbuswires. המודול הנ"ל הינו אסינכרוני.

כניסות: $R0, R1, R2, R3, R4, R5, R6, R7, g\_in, DIN, r\_out, g\_out$

יציאות:out

מודול הALU_unit:

משמש בתור יחידת חישוב לוגית עבור פעולות אריתמטיות כגון $add, addi, sub, subi$.

המודול הנ"ל הינו אסינכרוני.

כניסות: $alu\_in, regA, regB$

יציאות: $ALUout$

רגיסטרים:

כל הרגיסטרים הינם יחידות סינכרוניות.

רגיסטר G:

רגיסטר המחזיק את תוצאת פעולת החישוב של הALU :

כניסות : $g\ in\ , clock\ , alu\_out$

יציאות: RG

רגיסטר A

רגיסטר המחזיק את אחד הערכים המועברים לALU לצורך חישוב.

כניסות:$clk, ain, bus\ wires$

יציאות : a_out

רגיסטר IR

רגיסטר המחזיק את הפקודה המגיעה מכניסת הDIN ולהעביר אותה לתוך מכונת המצבים כאשר היא מקבל אות שמורה לה לעשות זאת מתוך מכונת המצבים.

כניסות : $DIN, IR_{in}, Clock$

יציאות: IR

רגיסטר $R0-7$

רגיסטרים באורך שמונה ביט שמטרתם לשמור מידע במערכת. כאשר הכניסה אליהם דלוקה (ביט במצב 1) הם נמצאים במצב של כתיבה, אחרת הם נמצאים במצב של זכרון (מחזיקים את המידע ללא יכולת שינוי).

כניסות : $buswires, rin[n], clock$, יציאה Rn כאשר $0 \leq n < 7$.

מודול מספר 1 : PROC

אותות כניסה:

- DIN : כניסת המידע , שנטענת IR כאשר ניתן לכתוב לרגיסטר IR , באורך תשע ביטים.
- Run : ביט שכאשר הוא פועל מסמן התחלה של ביצוע פקודה חדשה במעבד.
- CLK: שעון לטובת סנכרון המעבד.
- $Resetn$: ביט אשר קובע איפוס של המעגל כאשר הוא כבוי.

אותות יציאה:

- $Done$: אות באורך ביט אחד שמסמן סיום של פקודה במעבד.
- alu_out : אות באורך ביט אחד שמסמן לALU להתחיל לעבוד עקב כניסה של פקודת חישוב
- IR_{in} : אות באורך ביט אחד שמסמן למעבד לקרוא פקודה מרגיסטר IR.
- a_in אות שמאפשר למעבד לכתוב לרגיסטר a. כאשר הוא כבוי המידע הקיים ברגיסטר נשמר.
- g_in אות שמאפשר למעבד לכתוב לרגיסטר g. כאשר הוא כבוי המידע הקיים ברגיסטר נשמר.
- g_out אות שמורה לmux להעביר את תוצאת החישוב מהalu ל$buswires$

המודול מתפקד בתור הtop של הפרויקט וכן בתור מכונת המצבים שאחראית על מעבר בין מצבי המעבד.

הproc הינו יחידה סינכרונית , שפועלת בעליות שעון , השולטת בפעולות המעבד , הmux והalu באמצעות מכונת מצבים שבהתאם לטבלה ולפעולה הנתונה, מוציאה ביטים לmux ולalu על מנת לבצע פעולות לפי הסדר הנתון בטבלה ובזמן הרצוי.

הפקודה מתקבלת מרגיסטר IR שמוזן מכניסת הDIN , כאשר שלושת הביטים הראשונים נקראים בתור הפקודה , וששת הביטים האחרים של הפקודה, המייצגים את עוברים דרך מודול $dec3to8$ על מנת לקבל ייצוג 8 ביטים לכל אחד.

מכונת המצבים מורכבת משתי לולאות, אחת אחראית על מעבר ממצב למצב כתלות בביטים של $run, reset, done$ וכן דואגת להעביר את המכונה ממצב למצב לפי הסדר הרצוי, ולולאה נוספת המתפעלת את פעולות המעבד על ידי הדלקה וכיבוי ביטים , והקצאת רגיסטרים כנדרש. לטובת שליטה על פעולת המעבד, בכל שלב של מכונת המצבים כל האותות מוחזקים בערך (הערכים שלהם מצוינים בכל זמן מחזור) , על מנת להימנע ככל האפשר מהתנהגות בלתי צפויה.

$R_0 - R_7$: רגיסטרים באורך 8 ביטים לאחסון מידע , כאשר ניתן לכתוב אליהם ולקרוא מהם את המידע.

RA : רגיסטר ששומר מידע לפני כניסה לALU.

RG : רגיסטר ששומר את תוצאת החישוב של הALU ומעביר אותה לMUX כאשר מתקבל אות לעשות זאת.

$Tstep\_Q$: רגיסטר שמייצג את המצב הבא במכונת המצבים.

עקב היות המודול הנ"ל טופ של הפרויקט בנוסף, הוא כולל גם אינסטנסים של רגיסטרים עבור כל רגיסטר שצריך לממש, וכן אינסטנס של הmux ואינסטנס של הALU לצורך חיבור מכונת המצבים לmux ולalu.

מודול מספר 2 : mux_module

אותות כניסה :

- g_in - כניסת מידע באורך תשע ביט מרגיסטר g
- DIN – כניסת מידע באורך תשעה ביטים.
- $R0, R1, R2, R3, R4, R5, R6, R7$ - רגיסטרי מידע באורך שמונה ביט.

יציאות:

- r_out – רגיסטר יציאה באורך תשע ביט.
- g_out – ביט קריאה מרגיסטר g
- DIN_out – ב
- out – יציאה

מודול הmux הינו מודול אסינכרוני שתפקידו לבחור יציאה על פי אותות הכניסה שהוא מקבל ולהוציא אותה ל$buswires$.

הmux מקבל 10 כניסות שונות ,8 רגיסטרים באורך שמונה ביט, כניסת מידע באורך תשע ביט, כניסה ביט אחד g_in.

אם הmux מקבל מידע באחד הרגיסטרים , הוא יעביר ל$buswires$ את המידע הקיים ברגיסטר, אם יקבל כניסה מDIN יעביר אותה ישירות ל$buswires$, ואם יקבל אות g_in , הmux יעביר את התוצאה של הalu ל$buswires$.

מודול 3: ALU_unit

כניסות:

- alu_in - ביט הפעלת ALU
- A_val – ערך ראשון לחישוב.
- B_val – ערך שני לחישוב.

יציאות:

- $ALUout$

מודול הALU אחראי על ביצוע פעולות חישוב. הוא מקבל שני ערכים באורך תשע ביט, אחד מרגיסטר A במחזור T_1 והשני ישירות מה$buswires$ במחזור T_2 , ועל פי הפקודה שהוא מקבל מהFSM, הוא מבצע את פעולת החישוב ומוציא את התוצאה לרגיסטר g בזמן T_3. הALU מבצע את הפעולות $ADD, SUB, ADDI, SUBI$.

מודול 4: $dec3to8$

כניסות:

- W אות כניסה בעל שלוש ביטים המייצג רגיסטר.
- **En:** ביט הפעלה למודול.

יציאות:

- Y : קידוד 8 ביט לרגיסטר שהתקבל כקלט.

מודול ה $dec3to8$ הינו מודול אסינכרוני, המקבל מספר בין 3 ביטים שערכו נע בין 0 ל7 ומחזיר את הקידוד של הרגיסטר הרצוי בשמונה ביטים.

מודול 5: $regn$

כניסות:

- R: המידע שנכנס לתוך הרגיסטר, באורך שמונה ביט.

- Rin: ביט הפעלה של המודול, כאשר הוא פועל ניתן לכתוב לרגיסטר, כאשר הוא כבוי המידע נשמר.

- $Clock$: שעון לטובת סנכרון המודול.

יציאות:

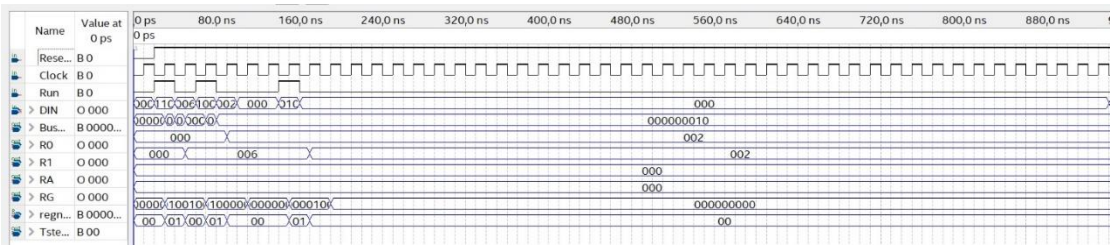
Q: מוצא הרגיסטר , המכיל את המידע, באורך שמונה ביט.

מודול ה$regn$ מייצר רגיסטר , שבעליית שעון דוגם מידע מהכניסה, ומעביר אותו ליציאה שלו , Q.

סימולציות

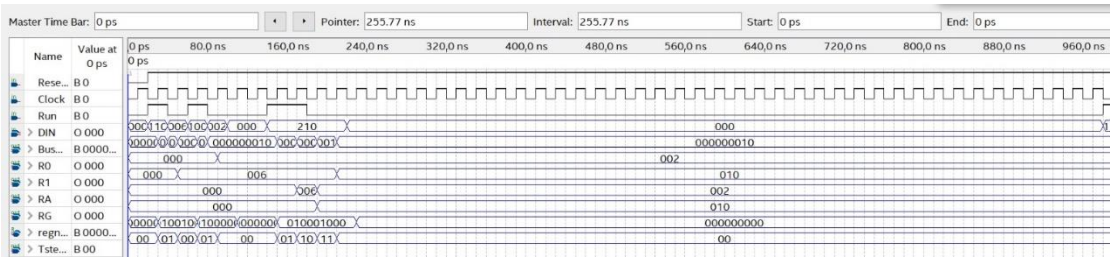
סימולציות של פעולות המעבד:

פקודות MOV ו- MOVI:



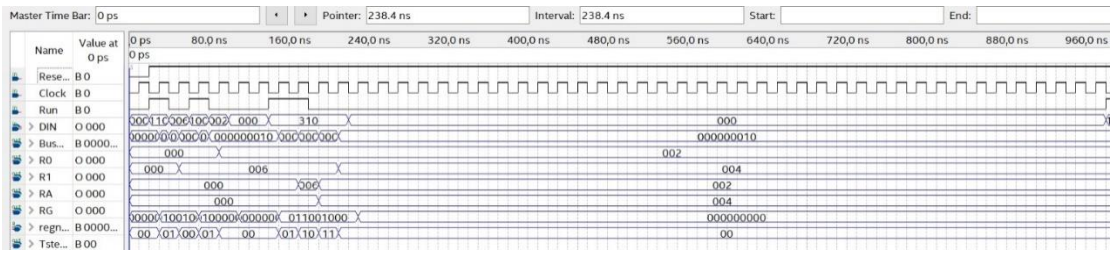
בסימולציה ניתן לראות פעילות תקינה של פקודת MOV המעבירה מידע מרגיסטר Y לרגיסטר X. ראשית איתחלנו את הערכים של הרגיסטרים בעזרת פקודת ה-MOVI אשר לוקחת שני מחזורי שעון ומעבירה את הערך ששמור ב-DIN לרגיסטר היעד. במקרה הזה בחרנו ב-R1 להיות רגיסטר היעד והערך השמור אצלו בהתחלה הוא 6. הערך השמור בריגסטר R0 הוא 2. לאחר הפעלת הפקודה, במחזור שעון השני, הערך שברגיסטר R0 יעבור לרגיסטר R1 ואכן כמו שניתן לראות הערך שנשמר ברגיסטר היעד הוא 6. ניתן לראות גם שכאשר ניתנת פקודת run הפעולה מתבצעת כמו שצריך.

פקודת ADD:



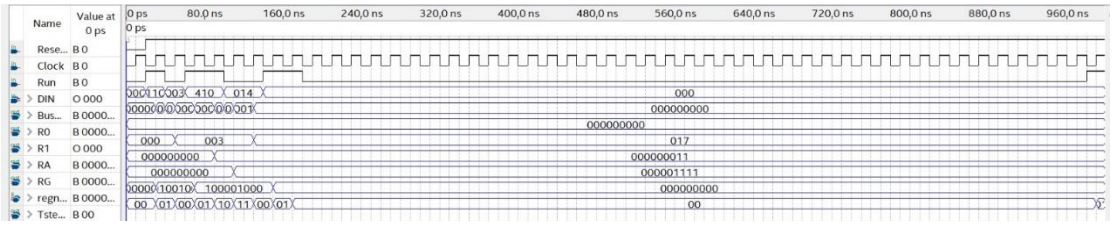
בדומה לסימולציה הקודמת, גם פה איתחלנו שני רגיסטרים R1 ו- R0 בעזרת פקודת MOVI שתיארנו לעיל כאשר $R0 = 2$ $R1 = 6$. לאחר האיתחול הפעלנו את פקודת ה-ADD אשר מחברת בין שני רגיסטרים ושומרת את התוצאה ברגיסטר היעד. פקודה זו מבוצעת על פני ארבעה מחזורי שעון. בסימולציה ניתן לראות כי בסיום הפקודה הערך שנשמר ברגיסטר R1 הוא הסכום של שני הרגיסטרים אותם הכנסנו. נציין כי ערכי הרגיסטרים מיוצגים בבסיס OCTAL. כך שרשום ברגיסטר היעד שהתוצאה היא 10 אך בבסיס עשרוני מקבלים 8 שזו אכן התוצאה.

פקודת SUB:



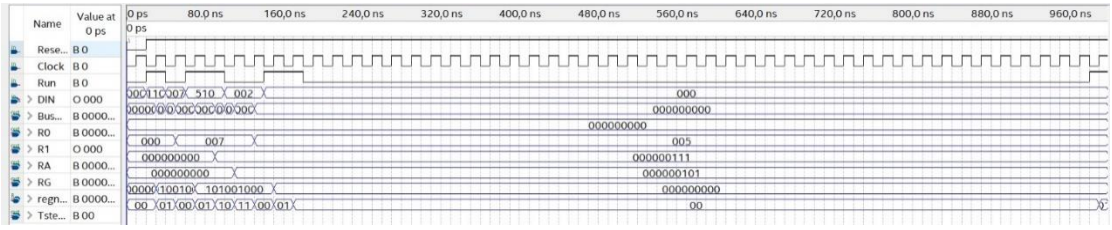
גם כאן איתחלנו את הרגיסטרים R0 ו-R1 עם אותם ערכים כמו קודם בעזרת פקודת MOVI. לאחר האיתחול הפעלנו את פקודת ה-SUB אשר לוקחת 4 מחזורי שעון ומחסרת בבין שני רגיסטרים. את התוצאה היא שומרת בגיסטר היעד R1. ואכן גם פה התוצאה שמקבלים מעידה על פעילות תקינה. בסוף הפעולה הערך שנשמר ב-R1 הוא 4 כפי שציפינו לקבל.

פקודת ADDI:



פקודה זו מחברת בין רגיסטר לבין IMMEDIATE השמור ב-DIN. לאחר איתחול רגיסטר R1 עם הערך 3 בעזרת פקודת MOVI הפעלנו את פקודת ADDI. ראשית נציין כי בדומה לפעולת ADD גם היא מבוצעת על פני 4 מחזורי שעון. במקרה של ADDI בשונה מ-ADD, במחזור שעון T2 אנו קוראים את ה- IMMEDIATE ששמור ב- DIN, במקום קריאה מרגיסטר Y. בחרנו במספר 12 (14 ב-OCTAL). ואכן ניתן לראות כי בסיום מחזור השעון הרביעי הערך של רגיסטר היעד שווה לסכום של 3 ו- 12 שהוא 15 (17 ב-OCTAL).

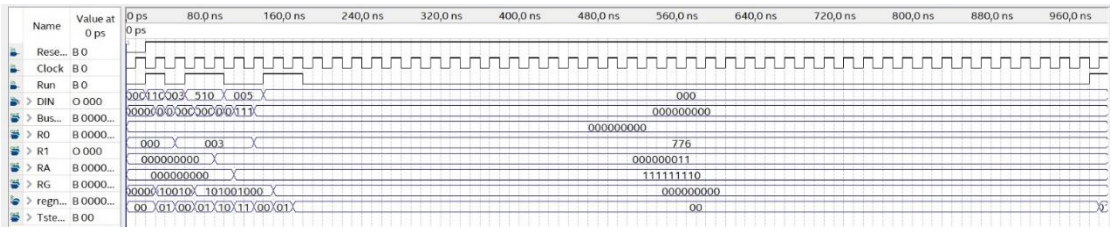
פקודת SUBI



פקודה זו מחסרת בין ערך של רגיסטר לבין IMMEDIATE השמור בDIN. ראשית איחתלנו את הערך ברגיסטר R1 (7) בעזרת פקודת MOVI. לאחר מכן הפעלנו את פקודת SUBI אשר לוקחת 4 מחזורי שעון. כפי שתואר בפקודת ADDI גם פה אנו קוראים במחזור שעון T2 את המידע מDIN (2 במקרה זה) ומבצעים את פעולת החיסור בעזרת ה-ALU. התוצאה נשמרת ברגיסטר היעד R1. ואכן קיבלנו כי התוצאה היא 5.

מקרי קצה:

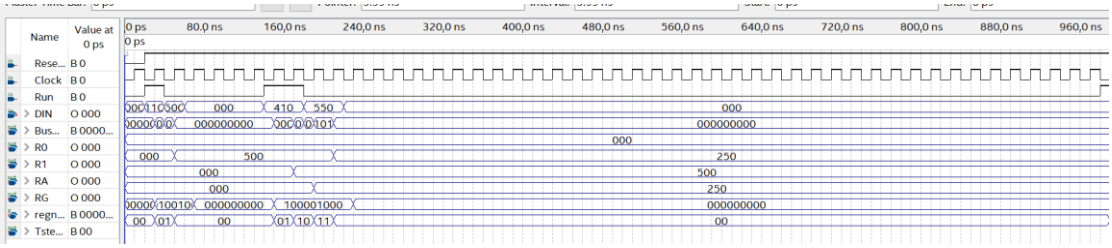
<u>פעולת SUB</u>



בסימולציה זו ניתן לראות את פעולת הפקודה SUBI בין הערך 3 ברגיסטר R1 לבין IMMEDIATE בעל הערך 5. כלומר הפעולה שמתבצעת היא $3 - 5 = -2$. ניתן לראות כי זהו ממש לא התוצאה אותה אנו מקבלים. המעבד שלנו לא מתוכנן להציג מספרים שליליים ובמקום מציג את הערך 510 (776 ב-OCTAL)

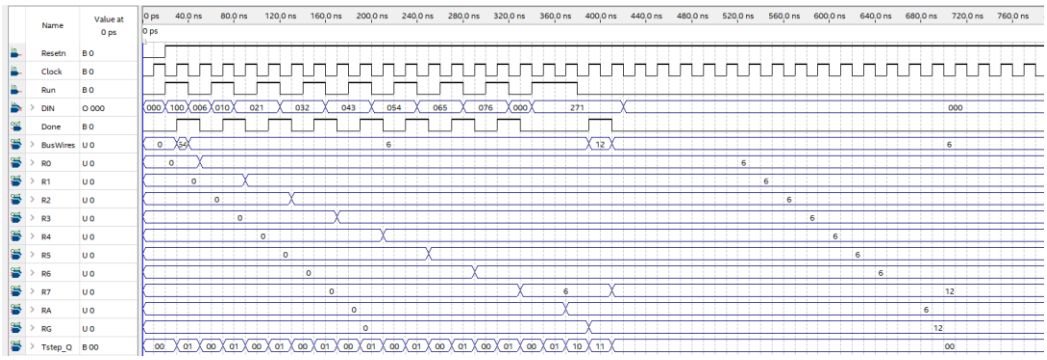
פעולת ADDI:

בסימולציה הבאה ניתן לראות כי חיבור בין רגיסטר המאותחל עם הערך 500 באמצעות MOVI בין IMMEDIATE בערך 550 נותן תוצאה שגויה – 250 (168 בעשרוני). זאת מאחר ואנו עוברים את כמו הביטים של הרגיסטרים, שהוא 9.



שימוש בכל הרגיסטרים :

נראה העברת ערך מ$R1$ עד ל $R7$ וחזרה ל$R1$ כדי להראות שכל הרגיסטרים עובדים.



ניתן לראות שהערך 6 מוכנס לרגיסטר 6, לאחר מכן מועבר מרגיסטר 0 לרגיסטר 1, וכך עד רגיסטר 7, ולאחר מכן נעשית פעולת חיבור בין רגיסטר 7 ל1 ששומרת את הערך 12 ב7. כמו כן ניתן לראות עבור פעולה add שהערך ברגיסטר A הוא 6 כצפוי לפעול החיבור, והערך ברגיסטר G הוא התוצאה של הALU. לכן ניתן להסיק כי כל הרגיסטרים עובדים בצורה תקינה.

מה לא בדקנו:

לא בדקנו פקודה שעובדת על כל הרגיסטרים ביחד (למשל פקודה שמזיזה את כל הערכים ברגיסטרים בו זמנית, או פקודה שמוחקת את המידע בכל הרגיסטרים, או פקודה שסוכמת את כל הרגיסטרים ביחד, וכו') . ניתן לבדוק את האפשרות הזו על ידי יצרת פקודת $mvall$ או $sumall$ או $erase\ all$.

כמו כן לא בדקנו פעולות לא חוקיות, כמו גישה לרגיסטר לא בזמן המחזור שלו ,או הפעלת הALU ללא ערך ברגיסטר A . ניתן לבדוק אפשרויות אלה על ידי הוספת מצבים במחזור T_1 שמיועדים לבדיקה של ערכים אלו.

כמו כן לא בדקנו את הטיימינג של המעבד עקב בעיות בתוכנה.

סיכום

המעבד אותו מימשנו הינו מסוג מולטי סייקל, כלומר, המעבד מבצע כל פקודה בשלבים כשכל שלב לוקח מחזור שעון אחד, בניגוד לסינגל סייקל שכל פקודה המבוצעת בו לוקחת מחזור שעון והזמן שלוקח תלוי בפקודה הארוכה ביותר.

יתרונות המולטי סייקל

- כל פקודה מתבצעת בזמן המחזור שלה ולא צריכה להיות באורך הפקודה הארוכה ביותר -יעיל יותר בהשוואה לסינגל סייקל
- יחידת השליטה שולחת אותות פר מצב ולא פר הוראה.
- אין כפילות חומרה
- זריז יותר מסינגל סייקל במקרים של ביצוע מספר של פקודות.

חסרונות

- דורש עוד רגיסטרים בהשוואה לסינגל סייקל כדי לשמור תוצאות של צעדים , ולכן מורכב יותר ליישום.

על מנת להפוך את המעבד לסינגל סייקל, נרצה שכל פקודה שהמעבד מבצע, תתבצע בזמן מחזור אחד בצורה אחידה.