

תרגיל בית רטוב מספר 3

קראו היטב את הוראות ההגשה בסעיף 3

נקודות יורדו למי שלא יבצע במדויק.

מתרגל אחראי לתרגיל: לאוניד עזריאל leonida@tx

שאלות בקשר לתרגיל יש להפנות ללאוניד דרך הפורום. בקשות מיוחדות יש לשלוח ללאוניד במייל


בקשות לדחייה ללא סיבה מוצדקת ידחו על הסף (ראו נוהל להגשה באיחור)

חשוב מאד:

אין להגיש שום חלק מודפס – את החלק היבש יש לכתוב במעבד תמלילים ולצרף לחלק הרטוב. ניתן להגיש כקובץ pdf, word או דומה.

1. הנחיות כלליות

מסמן שאלות שיש לענות עליהן במסמך (החלק היבש) 

מסמן חלק שיש לבצע בסימולטור. את תוצאת הסימולציה (waveform או מסך הסימולטור של קוד אסמבלי) יש לצרף לחלק היבש ולהסביר את התוצאות בצורה איכותית. waveform אפשר לקבל בעזרת צילום מסך. 

אין צורך לצרף את הקוד לחלק היבש אלא אם נאמר אחרת.

בתרגיל הזה תעבדו על מודל של מעבד RISC-V מסוג Multicycle כפי שנלמד בכיתה. בתרגיל תוסיפו פקודה אחת למודל זה.

2. הוספה של פקודות כפל למודל

בתיקיית RV שניתנה לכם עם התרגיל תמצאו מימוש של מעבד RISC-V כפי שמתואר בהרצאה 11. כמו כן התיקייה מכילה סביבת סימולציה למעבד. להלן רשימת הקבצים והסברים:

File	Description
rv_top.v	The top level of RISC-V
rv_dp.v	Data path section of RISC-V
rv_ctl.v	Control and state machine of RISC-V
rv_sim.v	Simulation top level
params.inc	Parameter definitions, included in Verilog files
imem.hex	Instruction memory image, read by simulation
dmem_init.hex	Data memory initial values, read by simulation
test.s	Assembly code used to generate machine code copied to the instruction memory image

המודל מממש תת-קבוצה של פקודות של RISC-V. להלן רשימת הפקודות הנתמכות:

Type	Commands
L	LW
S	SW
R	ADD, SUB, SLL, SLT, SLTU, XOR, SRL, SRA, OR, AND
B	BEQ
J	JAL

שימו לב שהקוד לא מוגן מפני פקודות לא נתמכות. ברוב המקרים אם תוזן פקודה לא נתמכת, מכונת המצבים תעבור לפקודה הבאה מייד אחרי ה-Decode. אך בחלק מהמקרים המכונה עשויה להתנהג באופן בלתי צפוי. בנוסף, קיים הבדל אחד בהשוואה למבנה שניתן בהרצאה. במודל Verilog הקידום של ה-PC ב-4 נעשה במקום מבלי להשתמש ב-ALU.

לפני שאתם מתחילים בביצוע התרגיל עברו על הקוד שקיבלתם וודאו שאתם מבינים אותו. הכניסו את קוד האסמבלי שנמצא ב-test.s לסימולטור <http://www.kvakil.me/venus>. השתמשו במקש Dump לקבלת קוד מכונה. וודאו שהקוד שקיבלתם זהה לתכולה של הקובץ imem.hex. הריצו את הטסט במודלים והעלו על דיאגרמת גלים את האותות המרכזיים כמו pc, ir, ALU inputs and output, memory interface¹.

2.1. עלייכם להוסיף פקודת mulw למעבד. למטרה הזאת תשתמשו בתשתית שבניתם בתרגיל 2. את החלק האריתמטי תשלבו ב-data path של המעבד (במקביל ל-ALU) ואת מכונת המצבים בבקר. שרטטו את התוספת הנדרשת לדיאגרמת המצבים של הבקר. ציינו את אותות הבקרה החדשים שנדרשים למימוש הפקודה. שימו לב שמתחשבים רק ב-32 סיביות הנמוכות, לכן מספר המצבים יכול להצטמצם.

2.2. כעת בצעו את השינויים הנדרשים בקוד Verilog. להלן רשימה חלקית של שינויים:

¹ כדי שמודלים ימצאו את קבצי hex. העתיקו את הקבצים ידנית לתיקייה של הפרוייקט שיצרתם.

- בקובץ params.inc הוסיפו סוג חדש של פקודות וכן אפשרות נוספת ל-Writeback input select. להלן הקידוד של פקודת mulw:

0000001	rs2	rs1	000	rd	0110011
---------	-----	-----	-----	----	---------

- בקובץ rv_dp.v הוסיפו את החלק האריתמטי של המכפל. הכניסו את החלק החדש יחדיו ליציאות A ו-B של register file. כמו כן הוסיפו עוד כניסה לבורר של כניסת DataD ל-Register file. הכניסה הזאת תקבל את המוצא של המכפל (32 סיביות תחתונות). מותר ומומלץ להשתמש במודול מתרגיל 2 כמושהו בקובץ נפרד תוך הצבתו במודול rv_dp.
- בקובץ rv_ctl.v הוסיפו את המצבים ואת אותות הבקרה החדשים.
- בקובץ rv_top.v הוסיפו את החיבורים החדשים.

2.3. בקובץ test.s main-ב הוסיפו בדיקה של מימוש הפקודה החדשה. השתמשו בפקודת lw לטעינה של ערכים מזיכרון מכתובות 8 ו-12 לרגיסטרים. בצעו כפל בין שני הערכים ואת התוצאה אחסנו בכתובת 16 בזיכרון. ייצרו קוד מכונה תוך שימוש בסימולטור <http://www.kvakil.me/venus> והעתיקו אותו ל-imem.hex. הריצו סימולציית ורילוג.

וודאו שבסוף ההרצה הקובץ dmem_out.hex מכיל את התוצאה בכתובת הנכונה (שורה 5). העלו על דיאגרמת גלים את האותות ir, pc, rst, clk ואת אותות הבקרה של הכפל. הוסיפו את הדיאגרמה לדו"ח.



חלוקת הציון

Sect	Grade
2.1	20
2.2	50
2.3	30

Total	100
-------	-----

3. הוראות הגשה

- 3.1. ההגשה בזוגות בלבד. ניתן לחפש בני זוג דרך פורום חיפוש שותפים. הגשה ללא בני זוג לא אישור מראש יגרור מאושרת מראש תגרור הורדה בציון של 10 נקודות
- 3.2. יש לארוז בקובץ zip בשם <id>.zip את כל קבצי הקוד (Verilog, assembly, memory images) וקובץ הטקסט של החלק היבש, כאשר id זהו מסי ת.ז. מלא של אחד מבני הזוג.

הערות:

- בתחילת כל תרגיל יש לכתוב שמות ו ת.ז. של כל אחד מהסטודנטים בטבלה כמו

בדוגמה:

שם 1	123456789
שם 2	987654321

- יש ליצור קובץ zip בודד עבור כל הקבצים. (ולא קובץ נפרד לכל אחד וישירות תחת ה zip וללא תתי תיקיות).
 - Zip - זה לא rar, לא 7z ולא שום תוכנת כיווץ אחרת.
 - אין להדפיס אף חלק בתרגיל. קובץ zip שיגיע ללא חלק יבש (קובץ טקסט) יגרור ציון 0 על התרגיל כולו.
- 3.3. הסימולציה תעבור בדיקה אוטומטית. אנו נריץ סימולציה על הקבצים שתספקו ולכן חשוב כי תשתמשו באותם שמות module המופיעים בתרגיל.
- יש להשאיר ללא שינוי את שמות הקבצים, את שמות ה modules - ואת שמות הפתחים (port - ים).
- 3.4. עליכם לעקוב אחרי הודעות אשר מתפרסמות באתר הקורס, הודעות אילו מחייבות. כל שאלה על התרגיל אשר איננה בקשה אישית צריכה להישאל דרך הפורום באתר הקורס.
- 3.5. אנא בידקו היטב את הקבצים לפני ההגשה. טענות מסוג "אבל בבית זה עבד נכון" לא תתקבלנה.
- קוד שלא מתקמפל יגרור הורדת נקודות מלאה של הסעיף

4. המלצות לתרגיל:

- 4.1. השתמשו ב-editor של modelsim או ב notepad++ או ב-vim
- 4.2. לא מומלץ לעבוד עם force
- 4.3. קודם חישבו ותכננו את המערכת ורק אח"כ התחילו לכתוב קוד. ככל שתקדישו יותר זמן לתכנון מוקדם - שלב המימוש יהיה קל יותר .
- 4.4. אל תחכו לרגע האחרון, שלב ה-debugging בד"כ ארוך ומסובך יותר משלב כתיבת הקוד.
- 4.5. לפני הגשה מומלץ ליצור תיקייה חדשה (נקייה) להעביר לשם את קבצי ה-source שלכם, ליצור את הפרוייקט מחדש ולהריץ שוב. בצורה זו תוכלו לוודא כי אכן הרצתם את הקבצים הכי עדכניים שלכם (הקבצים אותם אתם מגישים) ולא גירסה ישנה שקומפלה מזמן.