

תרגיל בית רטוב מספר 3

מערכות ספרתיות ומבנה מחשב

קראו היטב את הוראות ההגשה בסעיף 3.

נקודות יורדו למי שלא יבצע את ההוראות במדויק.


אחראית התרגיל נמרוד אדמוני nimrodadmoni@campus.technion.ac.il

שאלות בקשר לתרגיל יש להפנות לאחראי התרגיל דרך הפורום במודל. בקשות מיוחדות יש לשלוח לאחראי התרגיל במייל.

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


אין להגיש שום חלק מודפס – את החלק היבש יש לכתוב במעבד תמלילים (למשל: Word) ולצרף לחלק הרטוב. יש להגיש כקובץ pdf בלבד.

1. הנחיות כלליות

מסמן שאלות שיש לענות עליהן במסמך (החלק היבש). ניתן לענות גם באנגלית. 

לצורך שרטוט מעגלים עם שערים לוגיים בתרגיל, ניתן לשרטט ידנית ולסרוק, או להשתמש בתוכנה (למשל: <https://www.draw.io>). את החישובים יש להקליד.

בכל מקום בו נדרש לתכנן מימוש יעיל מבחינת מספר הרכיבים, אין צורך להגן מפני hazards.

מסמן חלק שיש לבצע בסימולטור. את תוצאת הסימולציה (waveform) יש לצרף לחלק היבש ולהסביר את התוצאות בצורה איכותית. ניתן לשמור Waveform בעזרת צילום מסך. ב-waveform יש להכיל את האותות הרלוונטיים לתרגיל (כניסות ויציאות) ובצילום להראות את הקטעים הרלוונטיים בזמן. יש לדאוג שהתמונות תהיינה ברורות. נקודות יורדו על צילומי מסך שאינם ברורים. 

יש לכתוב את קוד החומרה בשפת SystemVerilog בלבד וב-syntax שנלמד בסדנאות בלבד.

במקרה של קושי בפתרון הסעיפים היבשים או הרטובים, יש לפנות למתרגלי הסדנאות בפורום SystemVerilog וסימולציות ב-Moodle, או בשעות הקבלה שלהם.

בכל שאלה על **דרישות התרגיל**, כלומר קשיים בהבנת דרישות הסעיפים השונים בתרגיל, יש לפנות אל **אחראי התרגיל**, בפורום SystemVerilog וסימולציות ב-Moodle, או בשעת הקבלה.

בנוסף, ניתן להיעזר במסמך בעיות ב-ModelSim וב-SystemVerilog הנמצא באתר הקורס ([ModelSim_and_SystemVerilog_FAQ.pdf](#)).

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

בתרגיל זה יוצג מודל ב-SystemVerilog של מעבד RISC-V מסוג Multicycle כפי שנלמד בכיתה, אשר יורחב לתמיכה בפקודה חדשה.

2. הוספה של פקודת כפל למודל

בקבצי התרגיל שניתנו תוכלו למצוא מימוש של מעבד RISC-V כפי שמתואר בהרצאה 11. כמו-כן, התיקיה מכילה סביבת סימולציה למעבד. להלן רשימת הקבצים והסברים:

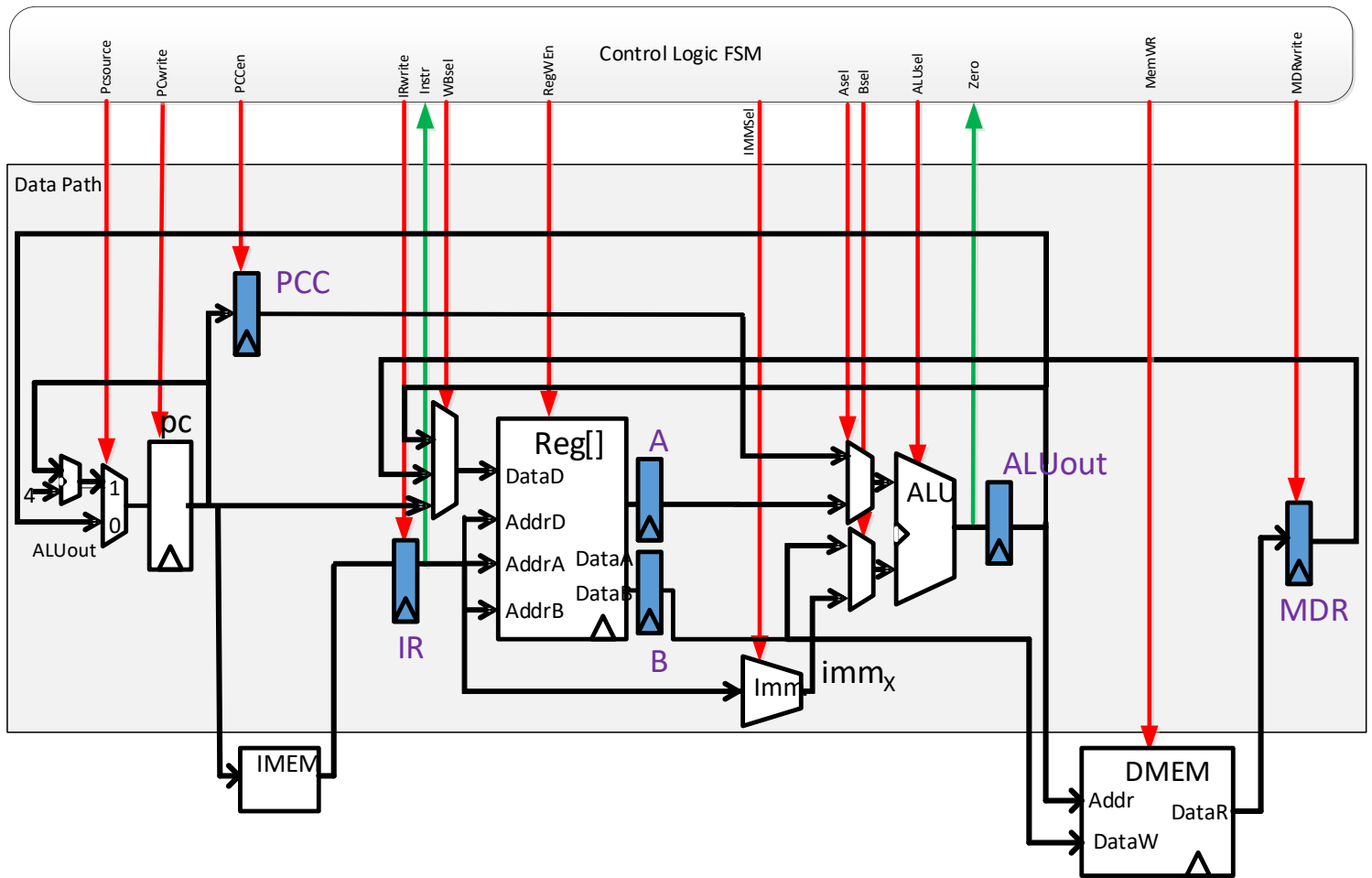
File	Description
rv_top.sv	The top-level hierarchy of RISC-V
rv_dp.sv	Data path section of RISC-V
rv_ctl.sv	Control and state machine of RISC-V
rv_sim.sv	Simulation top level
params.inc	Parameter definitions, included in SystemVerilog files
imem.hex	Instruction memory image, read by simulation
dmem_init.hex	Data memory initial values, read by simulation
test.s	Assembly code from which imem.hex is generated

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

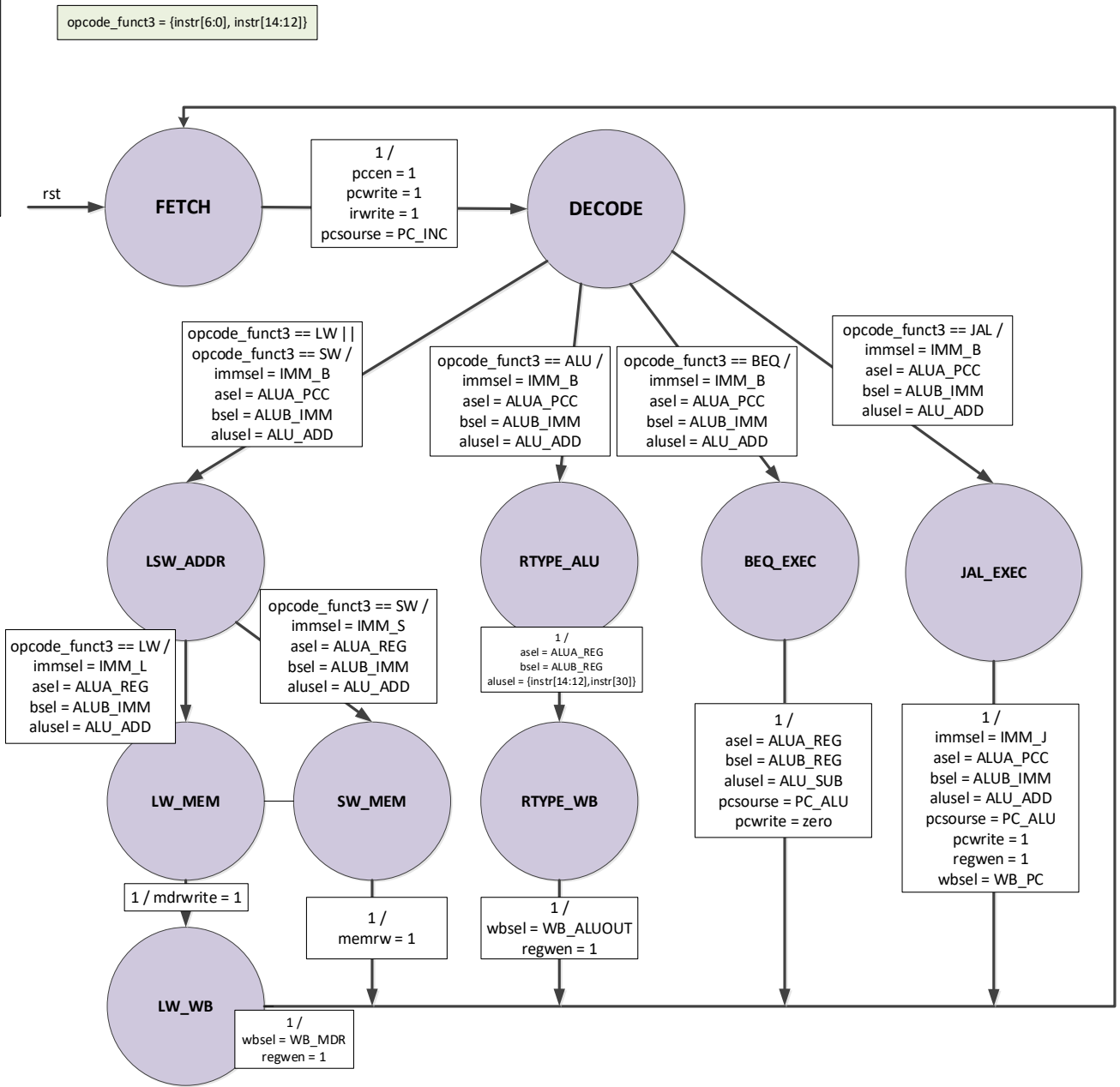
Type	Commands
L	LW
S	SW
R	ADD, SUB, SLL, SLT, SLTU, XOR, SRL, SRA, OR, AND
B	BEQ
J	JAL

שימו לב כי מודל המעבד הנתון אינו מוגן מפני פקודות לא נתמכות. ברוב המקרים אם תזן פקודה שאינה נתמכת, מכונת המצבים תעבור לפקודה הבאה מיד אחרי שלב ה-Decode. אך בחלק מהמקרים המכונה עשויה להתנהג באופן בלתי צפוי.

כאמור, המעבד הנתון בתרגיל אינו זהה לזה שנלמד בהרצאה. להלן ה-datapath ומכונת המצבים (מסוג Mealy) של המעבד בתרגיל (ניתן להשתמש ב-zoom כדי לראות את כל הפרטים):



Defaults:
 pcsource = PC_INC
 pcwrite = 0
 pccen = 0
 irwrite = 0
 wbsel = WB_PC
 regwen = 0
 immisel = IMM_B
 asel = ALUA_REG
 bsel = ALUB_REG
 alusel = ALU_ADD
 mdrwrite = 0
 memrw = 0



הקבצים imem.hex ו-dmem_init.hex, המייצגים את ה-instruction memory וה-data memory בהתאמה, בנויים בפורמט הבא: השורה הראשונה בקובץ מכילה את המידע שנמצא בבתים 0-3, השורה השנייה מכילה את המידע שנמצא בבתים 4-7, וכן הלאה. במידה ומופיעים פחות מ-4 בתים בשורה (כפי שניתן לראות ב-dmem_init.hex), הבתים החסרים הם ריפוד באפסים משמאל.

לפני שאתם מתחילים בביצוע התרגיל, עברו על הקוד שקיבלתם וודאו שאתם מבינים אותו. הכניסו את קוד האסמבלי שנמצא ב-test.s לסימולטור שבאתר: <http://www.kvakil.me/venus>. השתמשו בכפתור ה-Dump בסימולטור לקבלת קוד מכונה, וודאו שהקוד שקיבלתם זהה לתכולה של קובץ ה-imem.hex הנתון. הריצו את ה-test הנתון (rv_sim) ב-ModelSim. הוסיפו לדיאגרמת הגלים את האותות המרכזיים (למשל: pc, ir, ALU inputs + outputs, memory interface). ודאו שאתם מבינים את הנצפה בדיאגרמת הגלים. שימו לב שקוד האסמבלי הנתון נכנס ללולאה אינסופית בסוף הריצה. כשמריצים את הסימולציה על המעבד, היא מפסיקה בשני מקרים:

1. קבלת timeout.
2. כתיבה של ערך לזיכרון המצביע על סיום הקוד (שורות הפקודה תחת finish).

בתרגיל זה יהיה עליכם להוסיף למודל המעבד תמיכה בפקודת addi המבצעת פעולת חיבור עם intermediate. שימו לב שהפקודה שתוסיפו מתנהגת באופן שונה מהפקודה addi שנלמדה בהרצאות/תרגולים. בנוסף לחישוב תוצאת החיבור של רגיסטר כלשהו עם ערך של intermediate, היא גם מבצעת על תוצאת החיבור פעולת XOR עם הערך 0xffffffff (32 ביט שכולם מכילים 1). פורמט הפקודה הוא כפי שנלמד:

imm[11:0]	rs1	000	rd	0010011
-----------	-----	-----	----	---------

הפקודה מבצעת:

$$R[rd] = (R[rs1] + imm) \wedge 0xffffffff$$

2.1. הציעו מכונת מצבים חדשה עבור בקר המעבד. מכונת המצבים החדשה צריכה להיות מבוססת על מכונת המצבים הנתונה של המעבד, בתוספת המצבים הנדרשים עבור פעולת addi. שימו לב שפעולת ה-addi ופעולת ה-xor צריכות להיות מחושבות בתוך ה-ALU, במחזורים נפרדים. שרטטו דיאגרמת מצבים של מכונת המצבים החדשה. סמנו עליה את המצבים החדשים שהתווספו. ציירו על גבי שרטוט של המעבד מהם השינויים שהוספתם על-מנת לתמוך בפקודה החדשה.



2.2. כעת בצעו את השינויים הנדרשים בקוד ה-SystemVerilog של מודל המעבד. להלן רשימה חלקית של השינויים:



- בקובץ params.inc הוסיפו את הקידוד של הפקודה החדשה.
- בקובץ rv_ctl.sv הוסיפו למכונת המצבים הקיימת את המצבים ואת אותות הבקרה החדשים/שינויים באותות הבקרה הקיימים.
- בקובץ rv_dp.sv הוסיפו את השינויים הדרושים ב-datapath בהתאם לאותות הבקרה.

הנחיה כללית: שימו לב כי בכל קובץ קוד שקיבלתם יש לממש module אחד בלבד. אין לממש מספר modules בקובץ אחד, ואין להוסיף קבצי קוד חדשים בנוסף לאלו שקיבלתם.



2.3. בקובץ test.s ב-main הוסיפו בדיקה של מימוש הפקודה החדשה. השתמשו בפקודת lw לטעינה של ערך מזיכרון מכתובת 8. בצעו את הפקודה עם imm=12 (בבסיס דצימלי) והרגיסטר שאליו טענתם את הערך בכתובת 8, וכתבו את התוצאה לכתובת 16 בזיכרון. אין הגבלה לגבי שימוש ברגיסטרים ספציפיים בפקודות. יצרו את קוד המכונה בעזרת הסימולטור באתר: <http://www.kvakil.me/venus>, והעתיקו אותו לקובץ imem.hex. הריצו סימולציית SystemVerilog ע"י קימפול כל הקבצים בעלי סיומת .sv. (ניתן לעשות זאת באמצעות הפקודה vlog *.sv ואז הרצת הסימולציה (vsim rv_sim). הוסיפו לדיאגרמת הגלים את האותות: ir, pc, rst, clk. הוסיפו את הדיאגרמה לחלק היבש והסבירו אותה. שימו לב שלעתים עשויים להופיע קווים אדומים ('x') בסימולציה. בסימולציה זו, זה לא בהכרח מצביע על בעיה. ודאו שבסוף ההרצה הקובץ dmem_out.hex מכיל את התוצאה בכתובת הנכונה (השורה החמישית בקובץ). במידה והקובץ לא נוצר, משהו בהרצת הסימולציה היה לא תקין. הקוד שאחראי לסיום הסימולציה הוא הקוד שבא עם הקובץ test.s, לכן ודאו שהוספתם את הקוד שלכם במקום הנכון ולא שיניתם את הקוד הנתון.

חלוקת הציון

Sect	Grade
2.1	20
2.2	50
2.3	30

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

3. הוראות הגשה

- ההגשה בזוגות בלבד. ניתן לחפש בני זוג דרך פורום חיפוש שותפים. הגשה ללא בן זוג ללא אישור מראש תגרור הורדה בציון של 10 נקודות.
- יש להגיש את הקבצים הבאים (**ואותם בלבד**):
 - dmem_init.hex
 - imem.hex
 - params.inc
 - rv_ctl.sv
 - rv_dp.sv
 - rv_sim.sv
 - rv_top.sv
 - test.s
 - Dry.pdf
- יש לארוז את כל הקבצים הנ"ל (קבצי הקוד וקובץ התשובות של החלק היבש בפורמט pdf בלבד) בקובץ zip אחד, בשם **<id>.zip**, כאשר <id> זהו מס' ת.ז. מלא של אחד מבני הזוג.

- בתחילת קובץ התשובות לחלק היבש יש לכתוב שמות ות.ז. של כל אחד מהסטודנטים בטבלה כמו בדוגמה הבאה:

שם 1	123456789
שם 2	987654321

- שימו לב להגיש קובץ בפורמט zip בלבד! (לא rar, לא 7z ולא שום תוכנת כיוץ אחרת)
- אין להדפיס אף חלק בתרגיל. קובץ zip שיגיע ללא חלק יבש (קובץ pdf) יגרור ציון 0 על התרגיל כולו.
- הסימולציה תעבור בדיקה אוטומטית. אנו נריץ סימולציה על הקבצים שתספקו ולכן חשוב להשתמש באותם module-ים (שמות ו-port-ים) המופיעים בתרגיל.
- עליכם לעקוב אחרי הודעות אשר מתפרסמות באתר הקורס, הודעות אלו מחייבות.
- כל שאלה על התרגיל אשר איננה בקשה אישית צריכה להישאל דרך הפורום באתר הקורס.
- אנא בדקו היטב את הקבצים לפני ההגשה. טענות מסוג "אבל בבית זה עבד נכון" לא תתקבלנה. מומלץ לבדוק את תקינות הקוד על סביבה "נקייה" (למחוק את הספרייה work, ליצור אותה מחדש ולקמפל לתוכה מחדש את כל קבצי הקוד).
- קוד שלא מתקמפל יגרור הורדת נקודות מלאה של הסעיף.
- **שימו לב ל-Warnings שמתקבלים כפלט של הסימולטור. אזהרות יכולות להופיע גם בשלב ה-vlog וגם בשלב ה-vsime. נקודות יורדו על Warnings חמורים.**
- הגשה באיחור ללא אישור: על כל יום איחור יורדו 5 נקודות. הגשות באיחור מותרות עד שבוע מתאריך ההגשה.

4. המלצות לתרגיל:

- 4.1. מומלץ לערוך את קבצי הקוד בתוכנת [Notepad++](#).
- 4.2. חישבו ותכננו כל module לפני שאתם ניגשים לכתוב את הקוד שלו. ככל שתקדישו יותר זמן לתכנון מוקדם, כך שלב המימוש יהיה קל יותר.
- 4.3. אל תחכו לרגע האחרון. בד"כ שלב ה-debug ארוך ומסובך יותר משלב כתיבת הקוד.