דוקומנטציית פרויקט 2

בר ויצמן - 301393625

יהונתן בן הרוש - 206035883

הקדמה:

את הפרויקט שלנו ביצענו בהסתמכות על תכנית הTemplate שניתנה לנו ע"י סגל הקורס. תכנית זו בעצם הגדירה את רכיבי החומרה לבצע את הפקודות הבסיסיות. בנוסף, הוגדר בתוכנית זו שעון 4, בעל זמן מחזור של מילי-שנייה. בעזרת שעון זה, מימשנו את השעון בעל זמן המחזור של ה62.5 מילישניות.

בפרויקט זה החלטנו להשתמש במשתנים גלובליים, בכדי להקל על קוד הפרויקט, ובכדי שתהיה גישה נוחה מבלי שימוש במצביעים למשתנים השונים.

בקובץ זה נסביר את המימושים הפחות טריוויאלים שבוצעו בפרויקט שלנו.

משתנים וקבועים:

קבועים:

1. **XTAL_FRAQ –** הגדרת תדר הכרטיס : 80GHz
2. **MEM_LENGTH -** הגדרת הערך המקסימלי אותו ניתן לשמור במערך, 0xFFFF.

משתנים:

1. **Pause-** משתנה זה מקבל 0 כאשר אנו בריצה רגילה, 1 כאשר אנו בהשהיית התכנית, ו-2 כאשר אנו במצב single-step. הסבר המימוש בהמשך.
2. **Sw7 -** דגל ששומר האם סוויץ' 7 הועבר לאחרונה או לא, כלומר האם העברנו בין תוכניות.
3. **Pressed/swt_flag –** מערכים אשר שומרים את ערך הכפתור/סוויץ' באיטרציה הקודמת. המימוש מוסבר בהמשך.
4. **Imm_flag –** דגל ששומר האם קיים שימוש בקבוע בpc בו אנו נמצאים בתכנית האסמבלי. שימוש משתנה זה הוא בכדי לממש את שורת הINST.
5. **Func_flag –** דגל המחליט באיזה תוכנית אנחנו. הסבר נוסף בהמשך.
6. **Offset62 –** משתנה השומר את כמו המילי שניות שעברנו מאז ביצוע פקודת האסמבלי האחרונה
7. **Counter –** סופר את כמות הפעולות שבוצעו מאז הפעם האחרונה בה נלחץ reset.
8. **Mem_ind –** שורת הקוד בתכנית בה אנו נמצאים. אשר נעה בין 0 ל2047.
9. **Reg_ind/sec –** האינדקס בו אנו נמצאים בשורה הראשונה/השנייה בהצגת מערך הרגיסטרים. הסבר למה בוצע בפונקצייה זו בהמשך.

מערך הזיכרון:

בכדי לקצר זמני פעולה של הפרויקט ולחסוך בשימוש במצביעים, השתמשנו במערך אחד בגודל 4096 השומר אל תוכו את תוכן הזיכרון של שתי התוכניות. איברים הממוקמים באינדקסים 0-2047 שומרים את תכנית פיבונאצ'י, ואיברים באינדקסים 2048-4097 שומרים את תכנית תזוזת נורות הLED.

בכדי לבצע את השימוש הנ"ל, הגדרנו משתנה דגל בשם func_flag, שבאמצעותו, אנו ניגשים לאיבר במיקום ה - $pc + (func\_flag * 2048)$.

כאשר func_flag יהיה שווה לאפס, אנו בפיבונאצ'י, וכאשר יהיה שווה לאחד, אנו בתכנית הנורות.

לדוגמא, כאשר אנו בתוכנית האורות, ואנו מעוניינים בשורה השלישית של קוד התוכנית, משמע pc = 2, היא תהיה ממוקמת במערך הזיכרון במיקום 2050.

את מערך הזיכרון שמרנו בקובץ Header בשם Mem.h, למען אסתטיקת הקוד.

פונקציית הפסיקה:

בפונקציית הפסיקה מימשנו את השעון של 62.5 מילישניות בעזרת השוואה פשוטה בין currentTime שמונה את כמות מילי השניות שעברו מתחילת התוכנית, לoffset62 ששומר את כמות המילישניות שעברו מאז "תקתוק" השעון האחרון של השעון הדמיוני.

בנוסף, בדקנו בכל מילי שנייה האם בוצעה לחיצה או עזיבה של כפתור, ואם בוצעה, לבצע את תפקיד הלחיצה/עזיבה של הכפתור. את מימוש הפונקצייה שמטפלת בלחיצות ניתן לראות בהמשך הקובץ.

פונקציית switched:

בפרויקט זה, נאלצנו להבדיל בין מעבר של מצבים (העלאה או הורדה של סוויץ'), לבין הישארות במצב קיים.

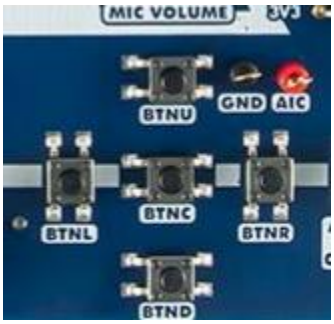
לדוגמא, כאשר אנו רוצים להשתמש בפונקציה המציגה על מסך הLCD את מערך הזיכרון, וברצוננו לבחור את האינדקס הראשוני שמוצג, אנו רוצים להתייחס אל זה כאל מעבר בין מצבים, כי אם היינו מסתכלים על זה כמצב, היינו "נתקעים" על האינדקס ההתחלתי ולא מסוגלים לדפדף בין איברי המערך.

לעומת זאת, כאשר אנו רוצים להציג את מערך הרגיסטרים על המסך, אנו רוצים להגדיר זאת כהישארות במצב.

בכדי להבדיל בין המקרים, השתמשנו בפונקציית switched, ה"מקבלת" את מערך הדגלים עבור סוויצ'ים, ואת ערך הסוויץ' הנוכחי. הפונקציה מחזירה את הערכים כמתואר בטבלה בעמוד הבא:

פונקציונליות	swt_flag[i]	SWT_GetValue(i)
הסוויץ' נשאר על מצב 0, לא התבצע שינוי מצב. (מחזיר 0)	0	0
הסוויץ' עלה, השתנה המצב. (מחזיר 1)	0	1
הסוויץ' ירד, השתנה המצב. (מחזיר 1-)	1	0
הסוויץ' נשאר על מצב 1, לא התבצע שינוי מצב. (מחזיר 0)	1	1

<u>פונקציית pressed:</u>



על פי ההגדרה של פרויקט הTemplate, כל עוד כפתור לחוץ, הדבר נחשב כלחיצה בודדת. דבר זה איננו פרקטי, מפני שלחיצה ארוכה תחשב כהרבה לחיצות, לדוגמא.

לכן, מימשנו את פונקציית pressed, ה"מקבלת" את מערך הדגלים של הכפתורים, ואת הערך של הכפתור ברגע הנתון, והשתמשנו במימוש הבא:

פונקציונאליות	btn_flag[i]	BTN_GetValue(i)
הכפתור לא היה לחוץ באיטרציה קודמת ועדיין לא לחוץ – לא נחשב כלחיצה. (מחזיר 0)	0	0
הכפתור נלחץ – לא נחשב עדיין כלחיצה. (מחזיר 0)	0	1
הכפתור נעזב – נחשב כלחיצה. (מחזיר 0)	1	0
הכפתור היה לחוץ באיטרציה קודמת ועדיין לחוץ – לא נחשב כלחיצה. (מחזיר 0)	1	1

*הנחנו, לפי הודעה בפורום הקורס, שלחיצת כפתור נקלטת כאשר הכפתור נעזב, ולא כאשר הוא נלחץ.

פונקציית function_executer:

תפקידה של פונקציה זו הוא בעצם לממש את פקודות האסמבלי הכתובות בהקסה-דצימלי, לפעולות.

השתמשנו בפונקציה מאוד דומה לפונקציה בה השתמשנו בפרויקט הקודם, תוך שינויים קלים בגדלי המערך.

בנוסף, הוספנו מונה ודגל.

המונה סופר כמה פעולות התבצעו מאז לחיצה על הreset בפעם האחרונה. מאותחל להיות אפס, וגדל באחד בכל כניסה לפונקציית הexecuter.

דגל הrun נדלק בפונקציית הפסיקה, בכל פעם ש62.5 מילישניות עוברות. בכך, אנו בעצם גורמים לתדירות פקודות האסמבלי המבוצעות להיות כרצוי. הדגל נכבה בכל פעם שאנו יוצאים מפונקציית הexecuter, ובכך אנו מוודאים כי פעולה אחת בלבד מבוצעת כל 62.5 מילישניות. (בעיקרון הדגל נדלק בממוצע כל 62.95 פעמים של ביצוע הפסיקה, כי לאחר מדידות באורך 10 דקות כל פעם, הגענו לכך שככה זה יותר מדויק).

```
if (run&&(pause!=1)) {
    function_executer();
    pc++;
    run = 0;
}
```

בפרויקט זה התבקשנו לממש השהייה של התוכניות. למען זאת, השתמשנו במשתנה pause. כאשר המשתנה מקבל את הערך 0, הפונקציות ירוצו כתלות בערך הדגל run הנ"ל. כאשר המשתנה מקבל 1, אנו בהשהייה, וגם אם דגל הריצה נדלק, פעולות לא מבוצעות.

בנוסף, התבקשנו לממש single step, כלומר, התקדמות בפקודה אחת בלבד בכל לחיצה על הכפתור הימני. מצב זה יכול לפעול רק אם אנו במצב השהייה. כאשר אנו רוצים להתקדם פעולה אחת, אנו משנים את pause ל2. כאשר דגל ההשהיה מקבל את הערך 2, תתבצע פעולה אחת, ובסוף פונקציית הexecuter, נחזיר את הדגל לערך 1, כלומר חזרה למצב השהייה.

פונקציית main:

בפונקציית הmain של הפרויקט, לאחר אתחול הרכיבים של הכרטיס, השתמשנו בלולאה אינסופית לביצוע הפעולות באופן מחזורי.

בהצגת מערך הרגיסטרים שכללנו מעט את הפונקציונליות. במקום להציג ערך אחד בלבד ממערך הרגיסטרים, החלטנו להציג שני רגיסטרים בו זמנית. ביצענו מספר התניות בכדי לוודא את כך שהדפדוף יהיה טבעי, גם במקרי הקצה.

בסוף הלולאה האינסופית, (אני אוהב את המשפט הזה) בדקנו האם התבצעה העברה של תוכנית. אם אכן הסוויץ' עלה או ירד, אנו נשנה את דגל הפונקציה func_flag, נאפס את הpc, ונאפס את IOReg[1] כדי שכשנעבור לפיבונאצ'י לא תהיה נורה שדולקת מתוכנית נורות הלד.

פונקציית האסמבלי של נורות הLED:

ראשית אתחלנו משתנים (את הנורה להיות נורה 0, וקבענו את הכיוון שמאלה, את מונה הלחיצות ל-0, ואת מונה הלחיצות שנלחצו גם ל-0).

בתחילת הלולאה החיצונית אנו משווים בין מונה הזמן שלנו ל-IOReg[0], אשר מונה את כמות הזמן מאז ה-reset האחרון שבוצע, כדי שנוכל לעשות את הלולאה באינטרוולי זמן קבועים. לאחר ההשוואה אנו נכנסים ללולאה הפנימית, שבודקת אם עברו 56 יחידות זמן של 62.5 מילי שניות. זאת מכיוון שלאחר ניסוי וטעיה, מצאנו את כמות הפעולות שצריך לבצע על מנת שכמות הזמן שלוקח ללולאה החיצונית לחזור על עצמה ("להזיז את האור"), יהיה חמש שניות.

לאחר היציאה מהלולאה הפנימית, ראשית נבדוק אם מאז הבדיקה האחרונה, נלחץ כפתור שהשנה כיוון מספר אי זוגי של פעמים, על מנת שנגיע אם צריך לשנות את כיוון התנועה של האור.

כעת נבדוק אם אנחנו בקצוות (נורה 0 או 7 דלוקה). זאת מכיוון שלא משנה כמה פעמים נלחץ הכפתור, כאשר אנחנו בקצה, יש כיוון תנועה אפשרי יחיד.

ולבסוף נשנה את IOReg[1] (מיקום הנורה הדלוקה) בהתאם לדגל הכיוון, ונחזור לראש הלולאה החיצונית.

מצב Programming:

לאור הרצון שלנו להפוך את הפרויקט שלנו ליותר אינטראקטיבי, החלטנו להותיר למשתמש את האפשרות לשנות בזמן השימוש בכרטיס, בכל אחת מהתוכניות, את אחד מהערכים שלה.

בתוכנית הפיבונאצ'י ניתן לשנות את האיבר אותו התוכנית תחשב (בין 0 ל-10), ובתוכנית הלדים את זמן ההארה של הנורה (מהמינימום האפשרי עד לכ-10 שניות)

על מנת להיכנס למצב תכנות, הכרטיס צריך להיות במצב pause וסוויץ' 3 צריך להיות דולק. כדי שלמשתמש תהיה אינדיקציה שכך שהוא במצב זה, מופיע על מסך ה-LCD "**Programming**", ואף ערכנו את הספרייה שאחראית על מסך ה-SSD, כדי שיציג "prog".

את ערכי התוכניות משנים על ידי לחיצה על הכפתורים למעלה ולמטה.