מבנה מחשבים ספרתיים
361.1.4191

Preparation report LAB1

בר קופפרשמיד 318912193

רותם ארביב 208065052

תאריך הגשה: 23/04/2025

שאלות חלק תיאורטי

1. רשום את תפקידם של הרגיסטרים $PxDIR, PxSEL, PxIN, PxOUT$

לכל פורט (PORT) יש 8 רגליים וניתן לתכנת כל פורט בצורה פרטנית וברמת הרגל הבודדת. כדי לתכנת PORT ישנם מספר רגיסטרים בגודל בית כאשר כל ביט מקנפג רגל נפרדת בPORT.

$PxDIR$ קובע את כיווניות רגל הבקר כאשר '1' מסמן כיווניות output ו'0' מסמן כיווניות input.

$PxOUT$ קובע את הערך הלוגי במוצא ברגל הבקר. כאשר '1' ערך המוצא הוא '1', כאשר '0' ערך המוצא הוא '0'.

$PxIN$ קריאת ערך המתח הלוגי ברגל הבקר. אם המתח הלוגי הוא '1' אז יהיה '1' ואם המתח הלוגי '0' אז יהיה '0'.

באופן כללי, רגלי הבקר יכולים לשמש למודולי חומרה שונים, לצורך כך ישנו רגיסטר $PxSEL$ המאפשר ברירה בין מודולי החומרה המשתמשים באותה רגל של הבקר. לבחירת רגל בקר למצב I/O ערך הביטים של $PxSEL$ יהיה שווה לערך ברירת המחדל שהוא '0'.

2. לאחר ביצוע RESET לבקר מהו מצב ברירת המחדל של הפורטים ומדוע?

בעת ביצוע RESET הבקר טוען את כתובת השמורה בכתובת המוגדרת כreset vector לרגיסטר הPC ולא מאפס את ערך הזיכרון – כלומר תוכן זיכרון הRAM נשמר.

ערך ברירת המחדל של $PxSEL$ הינו '0', לכן כל הפורטים מתוכנתים למצב I/O. ערך ברירת המחדל של $PxDIR$ הוא '0' כלומר input.

3. רשום את השלבים לצורך קינפוג PORT9 למצב I/O, כאשר מובאות בעלי אינדקס זוגי במצב output ומבואות בעלי אינדקס אי זוגי מצב input

1. קינפוג $P9SEL$ לערך '0' = $0x00$
2. קינפוג $P9DIR$ לערך הבא: $0x55 = 01010101$ (כך מבואות בעלי אינדקס זוגי במצב output ומבואות בעלי אינדקס אי זוגי במצב input).

אז בקוד אסמבלי:

$$BIC.B \quad \#0xFF, \&P9SEL$$

$$BIS.B \quad \#0x55, \&P9DIR$$

$$BIC.B \quad \#0xAA, \&P9DIR$$

4. כדי לייצר במוצא של פורט כלשהו גל ריבעי במחזור של 1ms, כמה מחזורי שעון MCLK נדרשים להשהייה עבור חלק של '1' באות הריבועי? נמק את תשובתך

נרצה למצוא כמה מחזורי שעון MCLK נדרשים להשהיית החלק ה'1' של הגל. אם מחזור הגל הריבועי הוא 1ms נרצה כי ההשהייה עבור החלק של ה'1' תהא 0.5ms. נזכור כי תדר השעון MCLK של הבקר הוא $2^{20}[Hz]$ לכן:

$$number\ of\ MCLK\ cycles@0.5ms = \left[\frac{\frac{1}{2}}{\frac{1}{2^{20}} \cdot 10^3}\right] = [524.228] = 524$$

5. הסבר מהי פסיקה ועל הצורך בה

פסיקה היא אות חשמלי המתקבל בCPU מרכיב חומרה (ניתן להפעלה גם ע"י תוכנה). פסיקה מאפשרת לשנות את סדר ביצוע הפקודות בתוכנית. בעת קבלת פסיקה המחשב משהה את ביצוע התוכנית באופן סדרתי וקופץ לכתובת כדי להפעיל רוטינת טיפול בפסיקה. בסיום הטיפול בפסיקה המחשב חוזר לתוכנית הרגילה וממשיך את התוכנית הסדרתית מאיפה שהוא עצר. פסיקה מאפשרת השהייה במקרים בהם יש לבחון תנאים או להמתין לאירועים חיצוניים כגון קלט מהמשתמש. בנוסף, פסיקה מאפשרת גמישות באופן ביצוע התוכנית, למשל ניתן לחלק משימה חישובית "כבדה" כמו חילוק למאיץ חומרה ולהמשיך בביצוע התוכנית הסדרתית שאינה קשורה עד קבלת פסיקה מהמאיץ שאומר שהוא סיים את החילוק. דוגמא לפסיקה המאפשרת פעילות תקינה של התוכנית היא RESET.

6. הסבר את הייתרון של שימוש בפסיקה (interrupt) לעומת תשאול (polling), מתי וכיצד נוכל לשלב בין השניים?

תשאול ממומש ע"י לולאה אינסופית שמבצעת בדיקה אם התקבל קלט על ידי המשתמש. הCPU יוצא מהלולאה לביצוע המשימה רק במקרה של קלט מהמשתמש. כלומר הCPU "תקוע" בלולאה עד לקבלת הקלט ולא פנוי לבצע משימות אחרות שאינן תלויות בקלט. לעומת זאת, בשימוש בפסיקה הCPU פנוי לביצוע פעולות אחרות ועוצר את פעולתו רק כאשר מתקבלת בקשה לפסיקה,

זהו ניצול מיטבי של משאבי המחשב. נרצה לשלב בין תשאול לפסיקה בעת הטיפול בפסיקות. כאשר תתקבל פסיקה, הCPU יפסיק את עבודתו ויקפוץ לטפל בה. לפני חזרת הCPU מהפסיקה נבצע תשאול על מנת לוודא שלא התקבלו פסיקות נוספות יחד/בזמן הטיפול הפסיקה בה טיפל כעת.

7. הסבר את שלושת סוגי הפסיקות ומה הצורך בכל סוג

באופן כללי ישנן פסיקות שאינן ניתנות למיסוך (דורשות טיפול מיידי שלא ניתן להעלם מהן למשל RESET) וישנן פסיקות הניתנות למיסוך.

ישנם 3 סוגי פסיקות:

- פסיקות חיצוניות (פסיקות א-סינכרוניות): פסיקות אלו נגרמות ע"י רכיב חומרה באופן שאינו תלוי בריצת התוכנית הנוכחית (לדוגמא לחיצה על לחצן).
- פסיקות פנימיות (פסיקות סינכרוניות): פסיקות אלו נגרמות ע"י חומרה או תוכנה במועד ידוע מראש עקב ביצוע פקודה מסויימת בתוכנית (לדוגמא טיימר פנימי).
- פסיקות תוכנה (פסיקות יזומות): פסיקות אלו נגרמות ע"י הדלקת דגל כלשהו בתוכנה (דימוי רכיב חומרה ע"י תוכנה).

8. הסבר את מושג אופני העבודה של הבקר, הסבר כל אופן הנפרד, ומתי תבחר להשתמש בו

אופני העבודה של הבקר:

1. **אופן עבודה תשאול (Polling Mode):** במצב תשאול, המיקרו-בקר מבצע בדיקה רציפה של מצב מערכת או רכיב אחר. לדוגמה, הבקר בודק אם יש נתונים להמשך עיבוד או אם יש צורך לבצע פעולה כלשהי, והוא עושה זאת בלולאה עד שהמצב משתנה. יתרון של מצב זה הוא הפשטות, אולם הוא לא יעיל במיוחד ומבזבז זמן, כי הבקר עוסק כל הזמן בבדיקות במקום לבצע פעולות אחרות. נבחר להשתמש במצב זה עבור פרויקטים קטנים שבהם לא נדרש ניהול פעולות מתקדמות.
2. **אופן עבודה עם פסיקות (Interrupt Mode):** במצב פסיקות, הבקר לא עוקב אחרי אירועים כל הזמן, אלא מבצע השהיית פעולתו בעת שמתרחש אירוע מסוים. כאשר האירוע מתרחש (כמו שינוי במצב פין קלט, סיום טיימר או נתונים שמתקבלים), הבקר עובר לפעולה שנקבעה לו בתגובה לפסיקה ואז חוזר להמשיך את הפעולה הקודמת. יתרון מצב זה הוא יעילות גבוהה, מאפשר ביצוע מטלות בזמן אמת, אך הוא מורכב יותר

למימוש ולתכנון. נבחר להשתמש במצב זה עבור מצבים בהם יש צורך בניהול פעולות בזמן אמת ובתגובה מיידית לאירועים.

3. **אופן עבודה במצב שינה (Low Power / Sleep Mode):** במצב שינה, הבקר מפסיק או מצמצם את פעילותו כך שהוא צורך כמות מינימלית של אנרגיה. מצב זה שימושי במיוחד במערכות ניידות או במערכות המתחברות למקורות כוח מוגבלים, כמו סוללות. יתרון המצב הוא חיסכון באנרגיה, והבקר יכול לפעול זמן רב יותר על מקור כוח מוגבל. עם זאת, יש צורך להחזיר את הבקר לפעולה לאחר הפסקה, והמצב לא מאפשר ביצוע פעולות רציפות. נבחר להשתמש במצב זה עבור מערכות ניידות או מבוססות סוללה.

9. רשום את השלבים כדי לקנפג את רגל P2.0 כך שבירידת מתח מ'1' ל'0' תתבצע בקשת פסיקה

1. קינפוג P2SEL[0] לערך '0' (כדי להגדיר מצב I/O)
2. קינפוג P2DIR[0] לערך '0' (כדי להגדיר אותו במצב input)
3. קינפוג P2IES[0] לערך '1' (כי נרצה בקשת פסיקה במעבר מ'1' ל'0')
4. קינפוג P2IE[0] לערך '1' (כדי לאפשר פסיקה)
5. קינפוג P2IFG[0] לערך '0' (כדי לנקות את הדגל אם מורם)

אז בקוד אסמבלי:

$$BIC.B \quad \#0x01, \&P2SEL$$

$$BIC.B \quad \#0x01, \&P2DIR$$

$$BIS.B \quad \#0x01, \&P2IES$$

$$BIS.B \quad \#0x01, \&P2IE$$

$$BIC.B \quad \#0x01, \&P2IFG$$

נשים לב כי יש לאפשר פסיקות גלובליות (ביט GIE) על מנת לאפשר קבלת פסיקות בפרט קבלת הפסיקה מרגל P2.0.

10. הסבר את כל אחת מפרדיגמות תכנות הבאות ואת ההבדל בניהן: Blocking, Non-blocking, Event driven, Interrupt driven

- **Blocking:** פרדיגמה זו מתארת קוד החוסם ביצוע של קוד אחר עד שלא הושלם במלואו.
- **Non-blocking:** פרדיגמה זו מתארת קוד המאפשר ביצוע של קוד אחר ללא המתנה לסיום הקוד המתבצע.
- **Event driven:** פרדיגמה זו מתארת קוד אשר מבצע את פעולתו בזמן שהוא ממתין לאירוע. ברגע שמזוהה אירוע הקוד (כאשר מתפנה, לא בהכרח במיידי) פונה לטפל בו.
- **Interrupt driven:** פרדיגמה זו מתארת קוד אשר עוצר את פעולתו כאשר מזוהה אירוע במיידי ופונה לטיפול האירוע על ידי אופרציות קבועות מראש ולפי תיעדוף בין האירועים (אם מתרחש יותר מאירוע אחד במקביל).

חלק מעשי

1. תכנון מערכת FSM

נדרשנו לתכנן מערכת FSM פשוטה עם המצבים הבאים:

<u>מצב 1 (בלחיצה על לחצן PB0)</u>: נדרש בלחיצה ראשונה להדליק על גבי 8 הלדים ספירה בינארית כלפי מעלה ובלחיצה שנייה ספירה בינארית כלפי מטה וחוזר חלילה (בכל פעם המנייה תתבצע מהיכן שהפסיקה בפעם האחרונה, על כן נדרש לשמור את ערך הכתיבה ללדים). הספירה תהיה מחזורית עם השהיה בין ערכי הספירה של 0.5sec משך זמן הפעולה יהיה 10 שניות. הערה: מצב אחר אינו רשאי "לחתוך" מצב זה טרם השלמת הביצוע המוגדר של המצב.

<u>מצב 2 (בלחיצה על לחצן PB1)</u>: נדרש להדליק לד בודד בדילוגים מימין לשמאל עם השהיה בין ערכי הספירה של 0.5sec משך זמן הפעולה יהיה 7 שניות (תוך שמירת ערך הכתיבה ללדים בחלוף הזמן, כך שבביצוע הבא של המצב הלד ימשיך לדלג מהיכן שהפסיק). הערה: מצב אחר אינו רשאי "לחתוך" מצב זה טרם השלמת הביצוע המוגדר של המצב.

<u>מצב 3 (בלחיצה על לחצן PB2)</u>: בלחיצה ראשונה המערכת מפיקה אות PWM במוצא רגל P2.7 בתדר 4kHz עם DutyCycle=75% ובלחיצה שנייה אות PWM במוצא רגל P2.7 בתדר 2kHz עם DutyCycle=25% וחוזר חלילה (אות המוצא נדרש להיות ברזולוציה מקסימאלית – ודאו זאת בעזרת שימוש בscope). הערה: מצב אחר רשאי "לחתוך" מצב זה (מאחר ופעולתו היא אינסופית) ובך לסיים אותו.

<u>מצב 0 ברירת מחדל (idle) :</u> הבקר מכבה את הלדים וחוזר למצב שינה(Mode Sleep).

גרף דיאגרמת הFSM שציירנו :