פתרון מועד א'

1. כזכור בבחינה תוקן מ-periodic ל-update.

רוב הסטודנטים ענו על השאלה נכון. היה צריך ב-

start\_70h\_peridic\_emulation(); לשמר את פסיקה 8 הנוכחית, להשתלט על פסיקה 8

רוטינת הטיפול החדשה של פסיקה 8 צריכה לבדוק אם חלפה שנייה, ואם כן לבצע

INT 70h

ולכתוב 1 לתוך US באוגר סטטוס C.

בדיקת חלוף שנייה היה אפשר לעשות ע"י מניה של פסיקות או לקרוא את אוגר 0 בפורט 71h ולבדוק אם השתנה מקריאה קודמת.

בכל מקרה היה צריך לקרוא ל-ISR המקורי או לכתוב 20h לתוך פורט 20h בכדי להודיע לחומרה שהפסיקה טופלה.

בשחזור פשוט משחזרים את פסיקה 8.

בסעיף 2 היה צריך לשנות את התדירות של פסיקה 8 ל-1000 ע"י כתובה של ערך מתאים (נניח 700( לתוך ה-latch של ה-PIT דרך פורט 43h ולעדכן את בדיקת חלוף הזמן, אם הסתמכה על תדירות.

טעויות נפוצות:

רוב הסטודנטים שכחו את עניין ה-US באוגר סטטוס C.

כמה שכחו לעדכן את עניין ההודעה לחומרה.

היו כמה ש"תקעו" את המעבד ברוטינה start שזה נוגד את רעיון הדמיה של פסיקה.

1. א. כמו בשאלות תאורטיות רבות היה יותר מתשובה נכונה אחת לסעיף הזה. Pri inv קורה כאשר יש הצטלבות נעילות בין תהליכים שוני עדיפות. השאלה הייתה מתי אין צורך להתמודד עם זה. אם אין נעילות כאלה (נניח רק תהליך בעל עדיפות גבוהה יכול לנעול) הבעיה לא קוראת. תשובה שלא התקבלה הייתה להציע פתרון לpri inv.

ב. ריבוי יישומים במערכת רגישה יכולה ליצור כמה בעיות אבל לרובם יכול להיות שיש פתרון למעט אחד: הרכיבים האסינכרוניים בקלט. אם 2 יישומים קרטיים מקבלים קלט באותו רגע, אי אפשר להבטיח זמן תגובה לשניהם. למשל במטוס מחשב אחד שגם יקבל קלט מטייס וגם יהיה בקר תפקוד של המנועים.

1. מתי בכל זאת אפשר להריץ 2 יישום קריטיים. יש כמה תשובות קבילות אבל הכללית ביותר לפי דעתי הוא שאם רק לאחד מהם יש רכיב אסינכרוני ולשני לא, אז זה אפשרי. דוגמא ליישום קריטי שאין לו רכיב אסינכרוני הוא רמזור.
2. בשאלה הזו היה צריך לממש רשימה מקושרת בעזרת חוצצים ממאגר, כאשר

 len =bufsize – sizeof(void \*)

הבתים הראשונים משמשים מערך של char ואילו sizeof(void \*) הבתים האחרונים של החוצץ משמש כמצביע ל-next כאלי כפי שאנחנו בדרך כלל חושבים. מאחר ומספר הבתים המשמשים למערך והמצביע לבא לא יכולים להיות מוגדרים ברשומהה, אלא צריך להשתמש באריתמטיקה של פוינטרים ו-cast –ים.

שליפה או כתיבה של איבר i מהמבנה היה כרך בסריקת הרשימה המקושרת, כל פעם מפחיתים len מ-i עד שאנחנו מוצאים i נהיה קטן מ-len, ואז ניגשים לאיבר len בחוצץ. בכדי לעבור לאיבר הבא במערך ניגשים למיקום len בחוצץ וקוראים משם sizeof(void \*) לתוך פוינטר.