דוקומנטציה לסימולטור

מטרת הסימולטור הינה לסמלץ את פעולת התוכנית לאחר שהassembler המיר את התוכנית בשפת assembly לקובץ בו כל שורה מתארת פעולה שעל הסימולטור לבצע או פקודת word. אותה הסימולטורת קורא כשורת קבועים.

מבני נתונים עיקריים:

הגדרנו מספר מערכים בהם נשתמש לאורך פעולת הסימולטור כדי לשמור את הנתונים:

א. MEM – מערך של הזכרון בגודל MAX_LEN_OF_FILE שהוגדר בפרויקט כ-4096, והטיפוס שלו הוא int. במערך זה התוכנית משתמשת ע"מ לשמור את זיכרון התכנית במהלך הפעולה ולהדפיס את הזיכרון הסופי לקובץ memout.

ב. R – מערך של הרגיסטרים בגודל 16. גם פה הטיפוס הוא int, לפי הגדרת גודל הרגיסטרים בפרויקט.

ג. Diskinmem- מטריצה של הזכרון בגודל NUM_SECTORS X NUM_WORDS_IN_SECTOR כפי שהוגדר בפרויקט כזיכרון של 128 סקטורים כאשר כל אחד מהם תופס 512 בתים (128 מילים), והטיפוס שלו הוא int. במערך זה התוכנית משתמשת ע"מ לשמור את זיכרון הדיסק הקשיח במהלך התכנית ועליו נבצע שינויים של כתיבה לדיסק הקשיח, ובסוף נדפיס אותו לתוך קובץ diskout.

ד. IOR- מערך של רגיסטרי החומרה בגודל 18. מתאר את רגיסטרי החומרה והוא מטיפוס unsigned int כי אין משמעות למספרים שליליים ברגיסטרים אלו והעבודה איתם דורשת הגדרה כזאת. כאשר רגיסטר מוגדר בפחות מ-32 ביטים, נתייחס רק לכמות הביטים ה-LSB שבה הוגדר בפרויקט. לדוגמה עבור irq0enabled ניקח רק את הביט ה-LSB .

התוכנית משתמשת בארבע פונקציות עיקריות על פי הסדר:

1. קריאת הקובץ memin לתוך מערך הזיכרון:

רצים על הקובץ לפי שורות. כל שורה ממירים ממחרוזת הכתובה במספר הקסאדצימלי וממירים את המחרוזת הזאת ל-int של 32 ביטים ומכניסים למקום המתאים במערך.

2. מעבר על כל הפעולות שתורגמו מהאסמבלר:

א. קוראים את ההוראה מהזיכרון הראשי mem לפי כתובת ה-PC (משתנה שהגדרנו כך שירוץ על מיקומי הכתובות בזיכרון).

ב. הפוינטר מתעדכן כל פעם למקום במערך לפי הPC שהוגדר בפעולה.

ג. מפצלים את ההוראה שכתובה במיקום הנוכחי בזיכרון למידע המתאים לפי opcode, rd, rs וכולי לפי הגדרת ההוראה בפרויקט.

ד. מדפיסים את ה trace של אותה פעולה.

ה. מפענחים את הפעולה ומבצעים אותה ע"י פונקציית ()Action.

ו. סופרים את מספר הפעולות שביצענו ע"י הוספת מחזור שעון למשתנה גלובלי מסוג unsigned long -TotalCycles, ומעדכנים את כל המשתנים הרלוונטים, למשל עדכון PC, עדכון שעון במידת הצורך וכולי.

ז. נמשיך לרוץ על הזכרון כל עוד לא הגענו ל-opcode=halt, או אם חרגנו מכתובת הזיכרון (כדי שהתכנית לא תקרוס)

3. <u>Action</u> – זוהי הפונקציה שמסווגת לפי סוג ה opcode איזה פעולה עלינו לבצע.

את הפעולות פיענחנו ע"י כך שהפרדנו את הביטים וביצענו בעזרת סדרת Switch-ים את הפעולה המתאימה.

נשים לב כי כאשר השתמשנו בהוראת קפיצה מכל סוג, דאגנו לעדכן את התכנית בכך שהקפצנו את ה-PC ע"י משתנה בוליאני, ככה שהתכנית לא תוסיף PC++ כאשר בצענו קפיצה בתוכנית ע"מ למנוע פספוס שורות בקובץ memin.

4. הדפסת הקבצים: לקובץ trace אנחנו מדפיסים בכל מחזור שעון את המידע הרלוונטי של הפעולה הנוכחית. ל-hwregtrace אנחנו מדפיסים כל פעם שקוראת פעולת out או in.

ל-leds, אנחנו מדפיסים כל פעם שישנה כתיבה לרגיסטר החומרה של הנורות.

ל-display, אנחנו מדפיסים כל פעם שישנה כתיבה לרגיסטר החומרה של התצוגה.

לכל שאר הקבצים אנחנו מדפיסים את המידע הרלוונטי בסוף התכנית.

5. Interrupts- בכל תחילת מחזור שעון אנחנו בודקים אם במחזור שעון הקודם זיהינו interrupt. אם זיהינו, והמעבד לא עסוק בפסיקה אחרת כרגע, אז נקפוץ לשגרת הפסיקה בהתאם לערך הנמצא ברגיסטר ששומר את מיקום קוד הפסיקה. אם אנחנו באמצע פסיקה, אזי לא נקפוץ לאף פסיקה. בדיקה זו מתבצעת ע"י משתנה בוליאני שכאשר אנחנו נכנסים לפסיקה אנחנו משנים אותו ל-true ואז אנחנו יודעים שאנחנו בפסיקה, וכאשר יש הוראת reti אנחנו הופכים אותו ל-false כדי שנדע שיצאנו מהפסיקה.