**פרויקט בארכיטקטורה של מחשבים – דוקומנטציה**

**מבנה התוכנית**

התוכנית בנויה משישה מודולים:

* **Main** – מודול זה הינו המודול הראשי של התוכנית, אחראי לאתחול הפונקציות שיממשו את פעולת המעבד, הרצת הלולאה הראשית שאחראית על הרצת המעבדים עד סיום עבודתם ויציאה מהתוכנית.
* **Core** – מודול האחראי על מימוש כל הפעולות שרצות עבור כל מעבד בנפרד.
* **Bus\_Mem** – מודול האחראי על מימוש לוגיקות הגישה לזיכרון הראשי ולבס.
* **Cache** – מודול האחראי על מימוש לוגיקות הגישה למטמון הנתונים.
* **Parser** – מודול האחראי על קריאת תוכן זיכרון ההוראות של הליבות.
* **Utils** – מודול זה אחראי על קריאה וכתיבה לקבצים, בנוסף מספר פונקציות עזר.

**מבני נתונים**

בתוכנית זו השתמשנו במבני נתונים, אשר כולם נמצאים בhard\_coded\_data.h, על מנת לייצג את הליבות, הצנרת והשלבים השונים בה, הבס, מטמון הנתונים, הזיכרון הראשי, תוך מניעת קוד מיותר לפיענוח ההוראות והמידע. נפרט על כל אחד מהם בקצרה:

* **Tsram\_Entry** – בלוק בזיכרון tsram.
* **Cache** – מטמון הנתונים של ליבה.
* **Address** – מבנה נתונים של הכתובות בזיכרון.
* **Instruction** – מכיל את ערכי כל הוראה, כפי שהובאה מזיכרון ההוראות.
* **Pipeline\_Stage** – מכיל ההוראות והפלטים עבור שלב בצנרת.
* **Statistics** – מכיל את ערכי הסטטיסטיקות עבור כל מעבד, אשר מתעדכנים בזמן הריצה.
* **MSI\_Bus** – מימוש הבס וניהול הבקשות אליו.
* **Core** – מבנה נתונים של הליבה המכיל את כל מבני הנתונים הקודמים.

**תיאור פעולה כללי**

לאחר אתחול כל הערכים, התוכנית מריצה בלולאה את הצנרות של כל הליבות, כאשר עבור כל ליבה מחושבת הצנרת בנפרד, ומעדכנת את הבס בסיום כל מחזור שעון, עד אשר כל הליבות מסיימות את פעולתם.

**פונקציות**

|  |
| --- |
| **מודול Core** |
| void initialize\_core\_statistics(core\* core) |
| אתחול מבנה הנתונים של הסטטיסטיקה עבור הליבה |
| void initialize\_core\_regs(core \*core) |
| אתחול הרגיסטרים של הליבה |
| void initialize\_core\_pipeline(core \*core) |
| אתחול ערכים לצנרת עבור הליבה |
| void initialize\_core(core \*core, char \*imem\_filename) |
| ריכוז הקריאות לפונקציות האתחול עבור הליבה ואתחול מבנה הנתונים שלה |
| bool check\_stage\_validity(core \*core, ePIPIELINE\_BUFFERS pipe\_buffer) |
| בדיקה האם השלב בצנרת הינו תקין |
| void check\_hazards(core\* core) |
| בדיקה האם יש hazard כבור הצנרת של הליבה |
| void fetch(core\* core) |
| מימוש שלב הIF בצנרת |
| void check\_branch(core\* core) |
| ביצוע branch resolution בצנרת במידה ורלוונטי |
| void decode(core\* core) |
| מימוש שלב הID בצנרת |
| void execute(core\* core) |
| מימוש שלב הEX בצנרת |
| void memory(core\* core, int \*main\_mem, int core\_num) |
| מימוש שלב הMEM בצנרת |
| void write\_back(core\* core) |
| מימוש שלב הWB בצנרת |
| void update\_stage\_buffers(core\* core) |
| קידום ערכי השלבים בצנרת, כתלות בstalls |
| void copy\_regs(core\* core) |
| העתקת ערכי הרגיסטרים |
| void simulate\_clock\_cycle(core\* core, FILE\* trace\_file, int \*main\_mem, int core\_num) |
| מימוש מחזור השעון בצנרת |
| bool all\_cores\_halt(core cores[CORES\_NUM]) |
| בדיקה האם כל הליבות קיבלו פקודת halt |
| **מודול Bus\_Mem** |
| void initialize\_main\_mem(int\* main\_mem, int\* memory\_request\_cycle) |
| אתחול הזיכרון הראשי |
| void initialize\_bus(msi\_bus \*bus) |
| אתחול הבס |
| void update\_bus(core \*cores, msi\_bus \*bus, int cycle, int\* next\_RR, int \*memory\_request\_cycle, int \*main\_mem) |
| עדכון הבס והזיכרון הראשי |
| **מודול Cache** |
| int tsram\_entry\_to\_line(tsram\_entry entry) |
| המרת כניסת tsram לשורה לשימוש עתידי |
| void initialize\_cache\_rams(cache \*core\_cache) |
| אתחול dsram וtsram עבור המטמון |
| void create\_bus\_request(core \*core, int core\_num, address bus\_addr, int request\_type, int data) |
| יצירת ערך בקשה, ע"פ הפרמטרים המתקבלים |
| bool mem\_block\_search(tsram\_entry \*tsram, int tsram\_index, int tag) |
| בדיקה האם יש hit או miss |
| int read\_mem(core \*core, int \*main\_mem, int core\_num) |
| מימוש פעולת lw |
| int write\_mem(core \*core, int core\_num) |
| מימוש פעולת sw |
| bool core\_snoop\_bus(core \*core, int core\_num, msi\_bus \*bus, bool \*shared\_flags) |
| קריאה של הבס עבור הליבה ופענוח הערכים הרלוונטים |
| **מודול Parser** |
| void remove\_spaces(char\* s) |
| הורדת הרווחים בקובץ טקסט. פעולה מסייעת לparse\_line\_to\_mem() |
| void parse\_line\_to\_mem(core\* core, char\* line\_buffer, int imem\_index) |
| תרגום שורה להוראה |
| void parse\_imem\_file(core\* core, char\* imem\_filename) |
| קריאת קובץ imem |
| **מודול Utils** |
| int address\_to\_integer(address addr) |
| תרגום כתובת מהקסדצימלי לדצימלי |
| void initialize\_args\_files(char\* args\_files[ARGS\_EXPECTED\_NUM - 1], int args\_num, char\* args\_values[]) |
| אתחול שדות הארגומנטים של הקבצים |
| int initialize\_array\_from\_file(char\* file\_name, int\* memory\_array, int max\_array\_size) |
| אתחול מערך ע"פ הנתונים שנקראים מקובץ הקלט |
| void format\_stage\_trace(bool valid, bool stalled, bool halt, char\* str, int num) |
| פעולה מסייעת לכתיבת קובץ coretrace |
| void write\_coretrace(core\* core, FILE\* trace\_file) |
| כתיבה לקובץ coretrace עבור ליבה |
| int write\_bustrace(msi\_bus\* bus, int cycle, char\* bustrace\_file) |
| כתיבה לקובץ bustrace |
| int write\_regout(core\* core, char\* regout\_file) |
| כתיבה לקובץ regout עבור ליבה |
| int write\_stats(core\* core, char\* stats\_file) |
| כתיבה של הסטטיסטיקות לקובץ stats עבור ליבה |
| int write\_dsram(core\* core, char\* dsram\_file) |
| כתיבה של תוכן הזיכרון dsram לקובץ עבור ליבה |
| int write\_tsram(core\* core, char\* tsram\_file) |
| כתיבה של תוכן הזיכרון tsram לקובץ עבור ליבה |
| int write\_memout(int main\_mem[MAIN\_MEM\_SIZE], char\* memout\_file) |
| כתיבה של תוכן הזיכרון הראשי בסוף התוכנית לקובץ memout |
| int write\_files(core\* cores, char\* args\_files[ARGS\_EXPECTED\_NUM - 1], int main\_mem[MAIN\_MEM\_SIZE]) |
| קריאה לכל פונקציות כתיבה לקבצים שאינם מתעדכנים, בסוף ריצת התוכנית |
| void update\_statistics(core\* core, int status) |
| עדכון מבנה הנתונים של הסטטיסטיקה על פי הנתון המתקבל |

**קבצי בדיקה**

* **Counter** –
* **Mulserial** –
* **Mulparallel** -