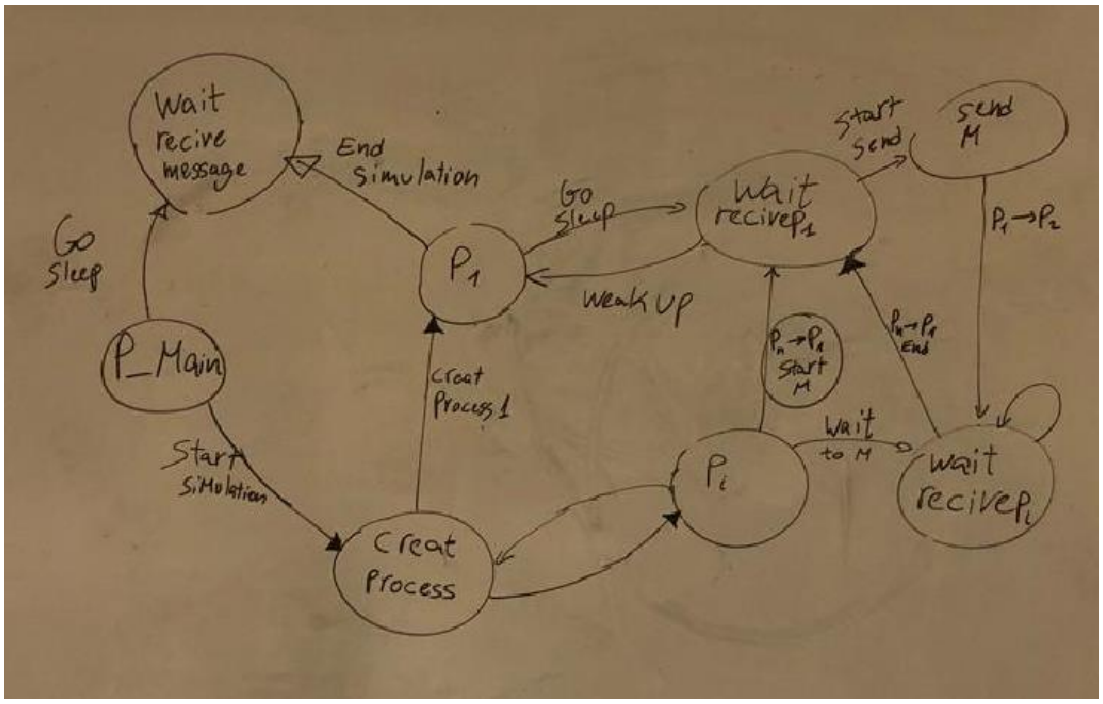
עבודה 5

מגיש : שחף זוהר תז - 205978000

<u>משימה ראשונה ring_parallel(N,M) –</u>

הסבר כללי על הפונקציה שממשתי, יצרתי מעגל שבו ה*MAIN* יצר תהליך 1 אשר יצר בצורה רקורסיבית N תהליכים אחד אחרי השני כלומר $P_n \leftarrow \cdots \leftarrow P_2 \leftarrow P_1$, עשיתי טריק כאשר רק תהליך הראשון משתמש ב $Tuple$ ושומר את ה*PID* של התהליך ה*MAIN* רק P_1 ,וכמובן מיד לאחר ניצור תבנית של הודעה שבה כל תהליך P_i יקבל תהליך P_1 שולח את התבנית של ההודעה לP_2 וכך הלאה כולם נכנסים למצב *wait* לאחר יצירת תהליך(P_i יוצר את P_{i+1} עד P_n).

תהליך P_n שולח ל P_1 הודעה שהוא סיים את המעגל, P_1 מקבל את ההודעה ומתחיל להעביר *m* הודעות. P_1 יודע מה הPid של P_2 וככה הלאה בנוסף ההודעה ה *m* היא הודעה מסוג מיוחד שנשלחת כאשר תהליך מקבל אותה הוא מעביר אותה קדימה לתהליך הבא במעגל ונהרג ככה עד תהליך P_1 שהוא שולח הודעה לתהליך ה *main* ואז נהרג.



<u>משימה השנייה ring_parallel (N,M) –</u>

במשימה הזאת יצרנו לולאה שבה כל איבר הוא בעצם ה*Process main* בהתחלה שלחנו לעצמנו את $\{1,0\}$, חזרנו על הודעה זו כאשר העלינו את ההודעה ככה דימינו את העברת ההודעה מ P_1 ל P_2 בהודעה כזאת $\{1,0\} \leftarrow \{2,0\}$ ככה עד $\{n,0\} \leftarrow$

כאשר נגיע ל0 נשלח הודעה כביכול לתהליך 1 שסיימנו טיול על הטבעת פעם אחת ונשלח לעצמינו שוב $\{1,1\} \leftarrow$ הודעה זו תשלח ותעלה עד$\{1,m\}$ בשלב זה אנחנו מסיימים את הסימולציה של המעגל שלנו כך דימינו את הטבעת הטורית.

	N=10,M=10	N=50, M=10	N=1000,M=10	N=10,M=50	N=10,M=100	N=10,M=1000
ring_parallel	45	280	7133	147	235	1452
ring_serial	22	61	1472	75	180	1722

- כל הנתונים הם בקנה מידה של Nano Seconds.
מסקנתי היא שהטבעת הטורית מהירה יותר בכל וריאציה של פרמטרים. ניתן להסביר זה מפני הפשטות של מימוש הקוד כאשר מדובר על תהליך בודד.אין צורך לבצע בדיקות היקפיות אשר באות להבדיל בין התהליכים השונים שכן בכל פעם מדובר על אותו תהליך.יכולנו לתלות את היעילות באופן המימוש, אך ניתן לראות שההבדלים במספרים הגדולים ניכרים ומשמעותיים.

משימה שלישית ring_parallel

- התהליך של main פותח את כל התהליכים
- תהליך main שולח לC (לצומת שנבחרה) הודעה שהוא הנבחר וקולט את ה Pid
- סידור כל המטריצה, כל תהליך יודע מי השכנים שלו ומה האינדקס שלו במטריצה {n^2-1}
- התהליך main שולח לC להתחיל סימולציה
- הצומת שנבחרה C מקבלת את ההודעה מהmain ויוצרת הודעה מהצורה $\{C_{index}, 1\}$ ומעבירה אותה לכל שכניה ברשת, ככה M הודעות כאלה עד $\{C_{index}, M\}$
- כל תהליך שמקבל הודעה כזאת דבר ראשון בודק האם ההודעה הזאת קיימת אצלו כבר במידה וקיימת הוא לא מעביר אותה לכל שכניו במידה ולא קיימת שומר אותה ומעביר אותה לכל שכניו ובנוסף יוצר הודעה עם האינדקס שלו אותה מספר הודעה $\{C_i, j\}$.
- כאשר תהליך C (הצומת שנבחרה על ידי ה user)מקבל כמות הודעות $M \cdot N^2$ מחזיר הודעה לmain שהוא סיים את הסימולציה וישירות לאחר מיכן הורג את עצמו
- תהליך main מקבל את ההודעה הזאת ומוציא הודעה לכלל התהליכים שנגמרה הסימולציה ושולח להם להרג.
- בסוף נמדד הזמנים ומוחזר.

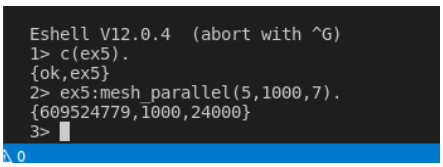
משימה רביעית mesh_serial

בדומה למה שבצעתי ב ring_serial ניסיתי ניסיתי לדמות רשת כפי שהוצגה כך שהמחשב מקבל ושולח לעצמו מספר הודעות.

	N=2,M=10	N=2,M=100	N=2,M=1000	N=3,M=10	N=3,M=100	N=3,M=1000
mesh_parallel	458	4346	211880	2846	46227	7207438
mesh_serial	91	197	2324	135	1700	14987

	N=4,M=10	N=4,M=100	N=4,M=1000	N=5,M=10	N=5,M=100	N=5,M=1000
mesh_parallel	4598	197531	111416517	26613	1685693	609524779
mesh_serial	428	4550	56859	1281	22328	107933

מחשב שלי קצת חלש אז לוקח זמן עבור ביצוע פעולה זו:



החלקים הטוריים והמקבילים:

- החלקים הטוריים ב mesh_parallel הינם החלקים בהם כל תהליך בודק האם ההודעה שקיבל התקבלה בעבר, ובהתאם מבצע את השליחה לשכניו.
- החלקים המקביליים ב mesh_parallel הינם "הסיבובים" בהם ההודעות עוברות בין תהליכים בצורה בלתי תלויה, כלומר מקבילית.
- כמובן שב mesh_serial הכל טורי, ולכן הזמנים יותר מהירים ההודעות עוברות באופן תורי דרך המיין באופן רקורסיבי ותלוי.