**Process:**

מופע הרצה של תכנית, מאז תחילתה עד סיומה, אם פתחתי 20 תכניות מאותו סוג יהיו 20 תהליכים שונים לכל אחד יש memory allocator משלו,

Memory allocator

**מחסן : HD/ SSD/ DOK**

שם שומר את כל המידע, יש שני סוגים:

* מידע שמטרתו לקריאה.
* Executable data: מכיל מידע שיכול להריץ תהליך מכיל מהתהליך את ה TEXT, DATA, וקצת STACK.

מערכת ההפעלה מזהה את השוני בין סוגי המידע.

כשיוצרים תהליך אנחנו ישירות מגדירים אותו כ executable data.

**CPU:**

* מעבד את כל התהליכים
* הוא עושה את כל החישובים XOR ים ...
* מכיל ALU : עושה את הפעולות האריתמטיות
* יש לו פסיק שעון שכל פעם שהוא עובר הוא משחרר את כל הפעולות,
* CPU מדבר עם ה RAM ולא הזיכרון עצמו

**RAM:**

מכיל MMU (מנהל את ה RAM) שמדבר עם ה CPU והוא מדבר עם כל הכניסות של ה RAM כל כניסה מכילה DATA ש RAM מביא דרך BUS שמחובר **למחסן.**

כאן גם שומר את המידע כשהתהליך בזמן קומפילציה.

**Cache:**

כאן שומרים DATA שהוא בעצמו קרוב ל CPU וכך יותר מהיר לקבל את המידע מאשר לדבר עם ה RAMוכך נחסוך פניות ל RAM זמן הפניות ל RAM יקר וזמן הפניות למחסן עוד יותר יקרה (בד"כ אם משתמשים במידע מסוים הרבה אז ה OS יעדיף לשמור אותו ב CACHE)ויש כמה רמות של CACHE\_ים.

**Process description:**

* האזור של ה DATA הוא המשתנים הגלובליים של התהליך
* המשתנים + מצביעים ב STACK
* אם עשינו מצביע גלובאלי ו CONST אז המידע הוא ב DATA ( read only –R/O-- ) אך המצביע יהיה ב stack .
* R/W – משתנה גלובאלי לא CONST שאפשר לשנות אותו בהמשך התכנית.
* BSS -- גלובליים לא CONST ים מאותחלים – אם יש לנו מערך בגודל 50000 וכל תא בגודל B2 (ז"א כל גודל הזיכרון ביחד הוא KB100) אז ה BSS ישמור SIZE\_T ובו\_B100000 וכשנאתחל אותו הוא יאתחל שם זיכרון בגודל KB100, עושים את זה כדי שלא נשמור אותו ב executable data ובעצם כשמיצריים את המערך הוא יוצר אותו על ה RAM ולא במחסן.

**ציור מהכיתה**

