הרצאה 2- מבנה מחשבים

מטרת הקורס שלנו היא לבנות CPU (Central Processing Unit). זוהי יחידת העיבוד המרכזית של המחשב ("המעבד"). יחידה זו נמצאת בתקשורת ישירה עם הזיכרון של המחשב, שם מאוחסנים הנתונים שעברו עיבוד במעבד.

כאשר מהנדסים מחשב, מנסים לייצר חומרה אשר יעילה הן מבחינת המהירות שלה והן מבחינת צריכת האנרגיה שלה. לעיתים הגדלת המהירות של המעבד באה על חשבון התחממות מהירה מדי של המעבד, על כן בשנים האחרונות לא חלה עלייה משמעותית במהירות של המעבדים שיוצרו.

אנו כותבים פקודות לחומרה של המחשב באמצעות שפת מכונה (אסמבלר). דוגמאות לשפות מכונה הן X86, שפה פופולרית של אינטל שפותחה בשנות ה-80 שבשימוש עד היום, ו MIPS. אנו נלמד אסמבלר במסגרת הקורס. כמו כן נלמד על האופן שבו המחשב מפרש את שפת המכונה.

עם תחילתו של עידן המחשוב, בשנות ה-50 וה-60, הייתה התלהבות רבה מהיכולות המתמטיות של המחשבים, ומהירות ביצוע החישובים. בתחום האקדמי נעשה ניסיון לשפר עוד יותר את המהירות ואת הפונקציונליות של המחשב.

עד שנות ה-70 כל חברה שייצרה את המעבדים למחשבים שמרה בסודיות מוחלטת על טכניקת הייצור שלה. אך בשנות ה-70 החליטה ממשלת ארצות הברית שעל מנת להגיע להישגים בתחום המחשבים, על החברות לחשוף את "סודות" הייצור שלהם, על מנת לאפשר דיבור אקדמי בתחום. בשל התנגדות החברות לחשיפה מוחלטת שתפגע בהכנסתם, הוחלט על הסדר שבו החברות יישלחו למוסדות אקדמיים מוצרים רק לאחר 5 שנים מעת יציאתם לשוק.

לפני שנות ה-80, מהנדסי המחשבים לא התחשבו בצרכיהם של אנשי התוכנה כאשר תכננו ובנו את המחשבים, אך לאחר שהתגלה כי זוהי גישה מוטעית, חל מהפך בתחום זה. המחשבים תוכננו ונבנו על מנת להתאים את עצמם לצרכיהם של אנשי התוכנה.

עד שנות ה-2000 מהירותם של המעבדים גדלה לאורך הזמן. אולם אז, לא הצליחו לייצר מחשבים של יותר מכ- 3GHz, מפני שבמהירויות גדולות יותר, כפי שנאמר, נוצר מצב שהמעבד התחמם במהירות ונשרף. לכן כיום מחפשים דרכים לחיסכון באנרגיה, כגון שימוש באנרגיה סולרית או בניית מחשבים מסיבים אופטיים.

בתחום החומרה, לאחרונה פותחו מחשבים מרובי ליבות (Multi Core), שבהם יותר ממעבד אחד, וכיום נעשה נסיון לנצל באופן מירבי את היכולת של מחשבים מסוג זה.

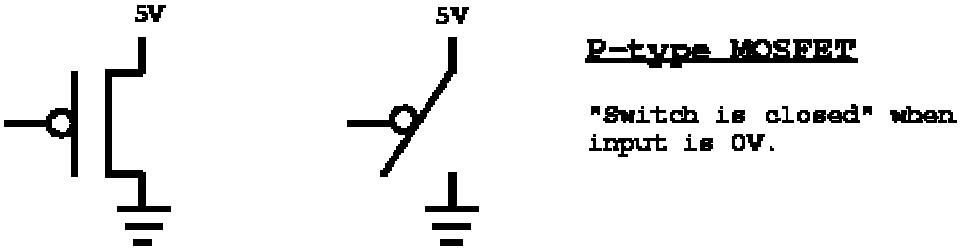
<u>תאימות</u>- על אף התכונות שמנינו כחשובות בעת ייצור מחשבים חדשים, הכרחי שמחשבים שמיוצרים היום יוכלו להפעיל תוכנות שכבר קיימות בשוק. לכן עדיין נעשה שימוש בשפת המכונה ה"מיושנת" X86. כמו כן רצוי שתוכנה חדשה תוכל לפעול גם על חומרה ישנה.

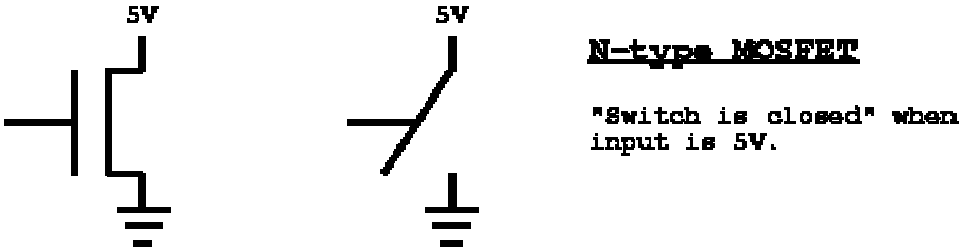
אנו מעריכים את המהירות של מעבדים שונים ע"פ התדר של ה-clock. תדר זה נמדד ב- MHz. התדר של ה- clock שווה ל 1/T, כאשר T נמדד ב nsec, זהו הזמן שבין שתי עליות שעון סמוכות.

שתי גישות לתכנות בשפת מכונה:
RISC- תכנות מקוצר, פחות פקודות במהירות גבוהה יותר (כגון X86).
CISC- תכנות בפקודות מסובכות ומתוחכמות במהירות נמוכהיותר (כגון MIPS).

לכאורה RISC היא גישה עדיפה כי היא יותר מהירה, שבה מתקיים חוק אמדל (Make the common case fast-כלומר יש ליעל קטע קוד המופיע לעיתים תכופות בתוכנית) אשר לא מתקיים ב CISC. אלא שלאחרונה ארכיטקטורת CISC פועלת במהירות זהה לRISC (או אפילו מהר יותר) .

<u>טרנזיסטורים-</u> טרנזיסטור הוא יחידה קטנה מאוד שעשויה מסיליקון, שתפקידה לשמש כמעין "מתג", כלומר להעביר או לחסום זרם חשמלי. צד שלישי מקבל או לא מקבל זרם חשמלי ולפיכך מחליט האם בטרנזיסטור יעבור זרם או לא.
אנו נלמד על טרנזיסטורים של CMOS שהם בשימוש כיום ע"י חברת אינטל.
נתעסק בשני סוגי טרנזיסטורים:
<u>P-type:</u>
כאשר מגיע מתח לטרנזיסטור המתג פתוח (הטרנזיסטור אינו מעביר זרם).
כאשר לא מגיע מתח לטרנזיסטור המתג סגור (הטרנזיסטור מעביר זרם).
<u>N-type:</u>
כאשר מגיע מתח לטרנזיסטור המתג סגור (הטרנזיסטור מעביר זרם).
כאשר לא מגיע מתח לטרנזיסטור המתג פתוח (הטרנזיסטור לא מעביר זרם).

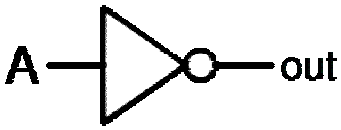




<u>שערים לוגיים:</u>

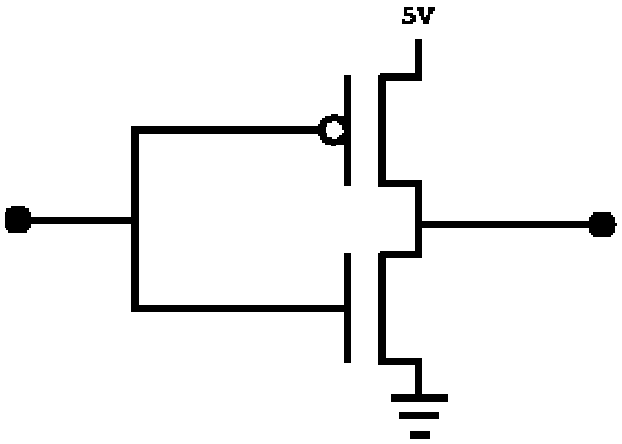
<u>שער NOT:</u>

<u>סימון:</u>



(אנו נקצר ונסמן בעיגול בלבד)

מימוש:

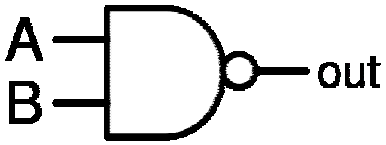


טבלת אמת של שער NOT:

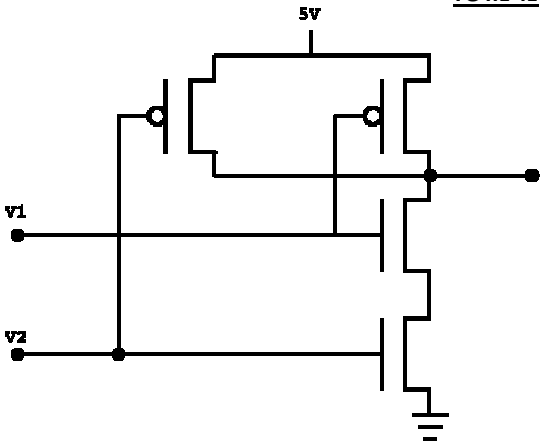
IN	OUT
0	1
1	0

שער NAND:

סימון:



מימוש:

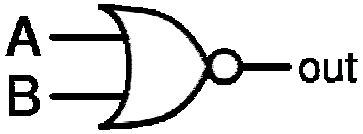


טבלת אמת של שער NAND:

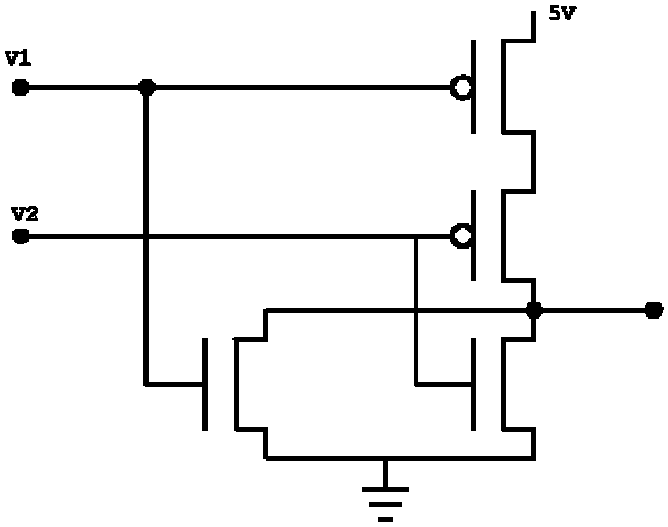
A NAND B	B	A
1	0	0
1	1	0
1	0	1
0	1	1

שער NOR:

סימון:



מימוש:



טבלת אמת של שער NOR:

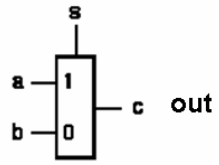
A NOR B	B	A
1	0	0
0	1	0
0	0	1
0	1	1

- מימוש של שער AND נעשה באמצעות חיבור של שער NAND לשער NOT.
- מימוש של שער OR נעשה באמצעות חיבור של שער NOR לשער NOT.
- שער NAND הוא מערכת אוניברסלית- באמצעותו בלבד ניתן לייצר את כל הפונקציות הבינאריות.

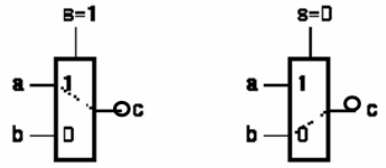
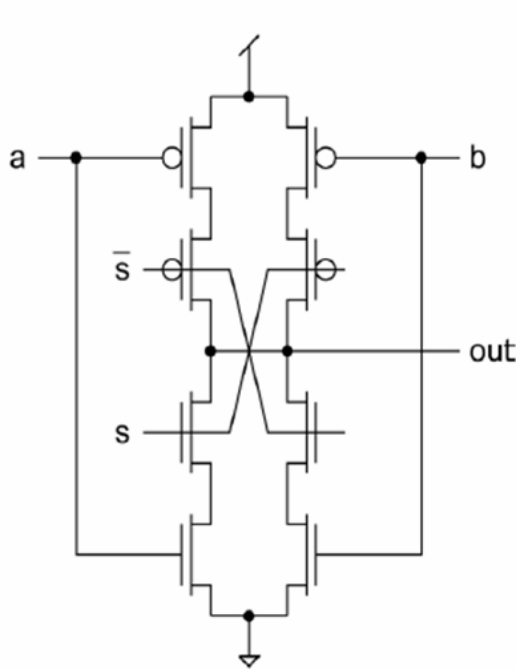
MULTIPLEXORS:

רכיב בעל כניסות מרובות, אשר בוחר איזה מהכניסות תעבור ליציאה.
לדוגמא הMULTIPLEXORS שהוסבר בכיתה הוא

Multiplexor



If s=1 then $c=\bar{a}$ else $c=\bar{b}$



מימושו: