

תוכן העניינים:

2	מבנה המחשב ותכן לוגי
2	המעבד הרב-מחזורי
2	מבנה המעבד הרב-מחזורי:
2	סיכום כללי:

מבנה המחשב ותכן לוגי

המעבד הרב-מחזורי

מבנה המעבד הרב-מחזורי:

סיכום כללי:

יתרונות של מעבד חד-מחזורי:

- מודל פשוט וקל להבנה ברמה הלימודית.

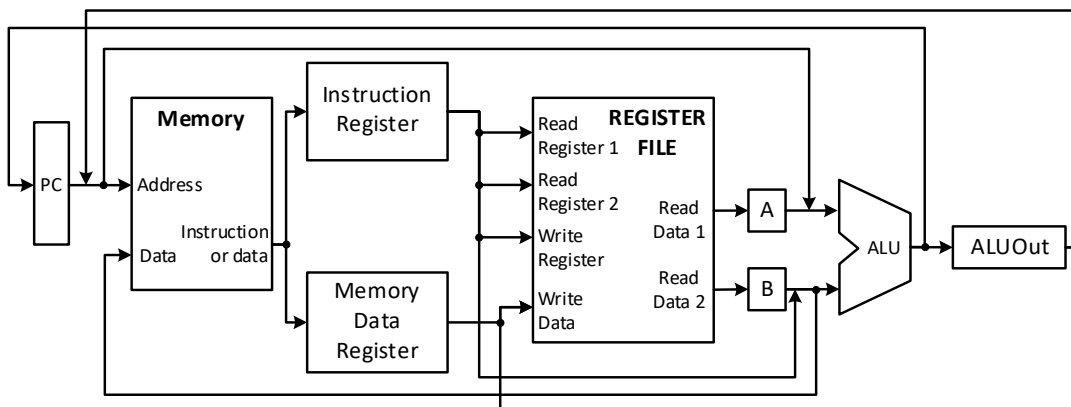
חסרונות של מעבד חד-מחזורי:

- ביצועי המעבד הם נמוכים מכיוון שזמן מחזור חייב להיות מותאם לפקודה הארוכה ביותר והיא lw. נוצר מצב שבו כל פקודה שאורכת פחות זמן משל lw מסתכמת בזמן. בזמן.
- ניתן להשתמש בכל רכיב חומרתי בדיוק פעם אחת בפעימת שעות ולכן יש צורך בשיכפול של יחידות חומרה פונקציונאליות.

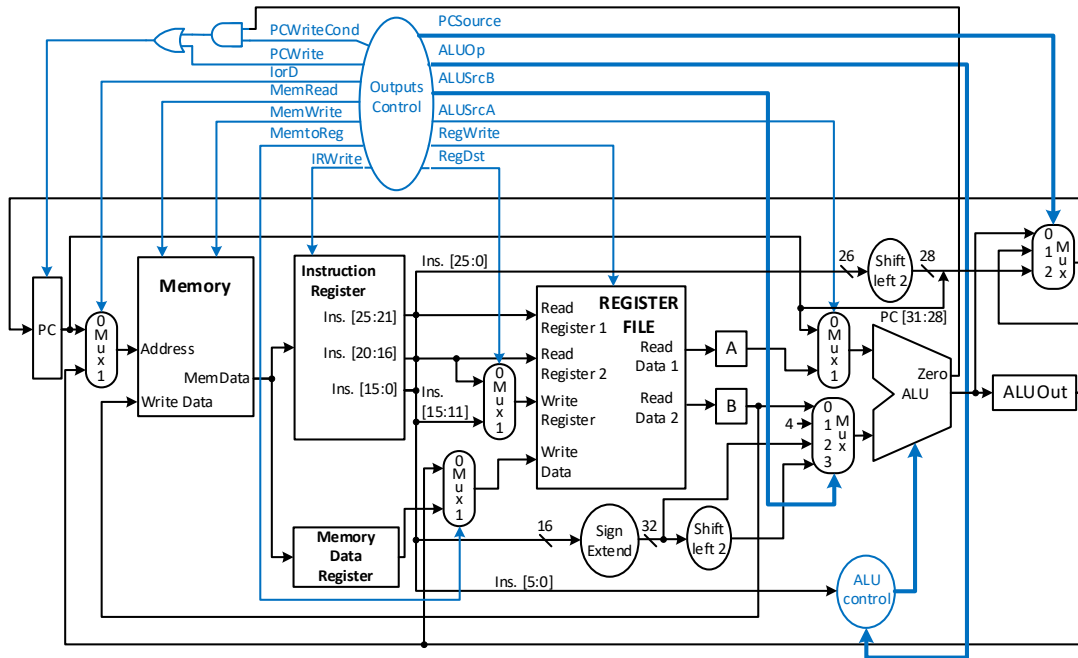
מעבר למעבד רב-מחזורי (Multi-Cycle Processor):

במקום להתייחס לביצוע של פקודה במשך מחזור שעות שלם, נשבור את ביצוע הפקודה למספר שלבים, המוגדרים לפי יחידות החומרה הפונקציונאליות של המעבד כפי שכתבנו מקודם.

שינוי בנתיב הנתונים:



מבנה המעבד המלא:



טבלת קווי בקרה בסיבית אחת:

Signal Name	Reset (0)	Set (1)
RegDst	The register destination is <i>rt</i> (Ins.[20:16]).	The register destination is <i>rd</i> (Ins.[15:11]).
RegWrite	No write is allowed	The content in <i>Write Register</i> is replaced with the new value from ALU or MDR.
ALUSrcA	First ALU operand is PC.	First ALU Operand is Register A.
MemRead	No read is allowed	The content of memory at <i>Address</i> is fed through <i>MemData</i> data-line to IR register.
MemWrite	No write is allowed	The content in <i>Address</i> is replaced by the value in <i>Write Data</i> line.
MemtoReg	Content to <i>Write Back</i> in the Register File is from ALUOut.	Content fed to <i>Write Back</i> in the Register File is from Memory Data Register.
IorD	The PC provides the address content	The ALUOut provides the address content.
IRWrite	No Action is performed	The output from memory is written into IR.
PCWrite	No Action is performed	PC is written, the source depends on PCSource.
PCWriteCond	No Action is performed	PC is written if zero bit is active.

טבלת קווי הבקרה בשתי סיביות:

Signal Name	Value	Meaning
ALUOp	00	ALU performs Addition.
	01	ALU performs Subtraction.
	10	The <i>funct</i> field determines the ALU operation.
ALUSrcB	00	The second ALU operand comes from B register.
	01	The second ALU operand is a constant 4.
	10	The second ALU operand is the sign extended lower 16 bits from the IR.
	11	The second ALU operand is the IR sign extended and shifted left by 2 bits.
PCSource	00	PC gets the value of PC + 4 from the ALUOut.
	01	PC gets the ALUOut content from branch commands.
	10	PC gets the jump command address.

סיכום שלבי ביצוע פקודות של מעבד רב-מחזורי:

Step Name	Action for R-type instructions	Action for memory reference instructions	Action for branches	Action for jumps
Instruction Fetch	IR <= Memory[PC] PC <= PC + 4			
Instruction decode / Register fetch	A <= Reg[IR[25:21]] B <= Reg[IR[20:16]] ALUOut <= PC + (sign-extend(IR[15:0]) << 2)			
Execution, address computation/branch/jump completion	ALUOut <= A op B	ALUOut <= A + sign-extend (IR[15:0])	if (A == B) PC <= ALUOut;	PC <= {PC[31:28], IR[25:0], 2'b00}
Memory access or R-type completion	Reg[IR[15:11]] <= ALUOut	MDR <= Memory[ALUOut] or: Memory[ALUOut] <= B		
Memory read completion		Reg[IR[20:16]] <= MDR		

השוואה בין מחזורי השעון לביצוע פקודות:

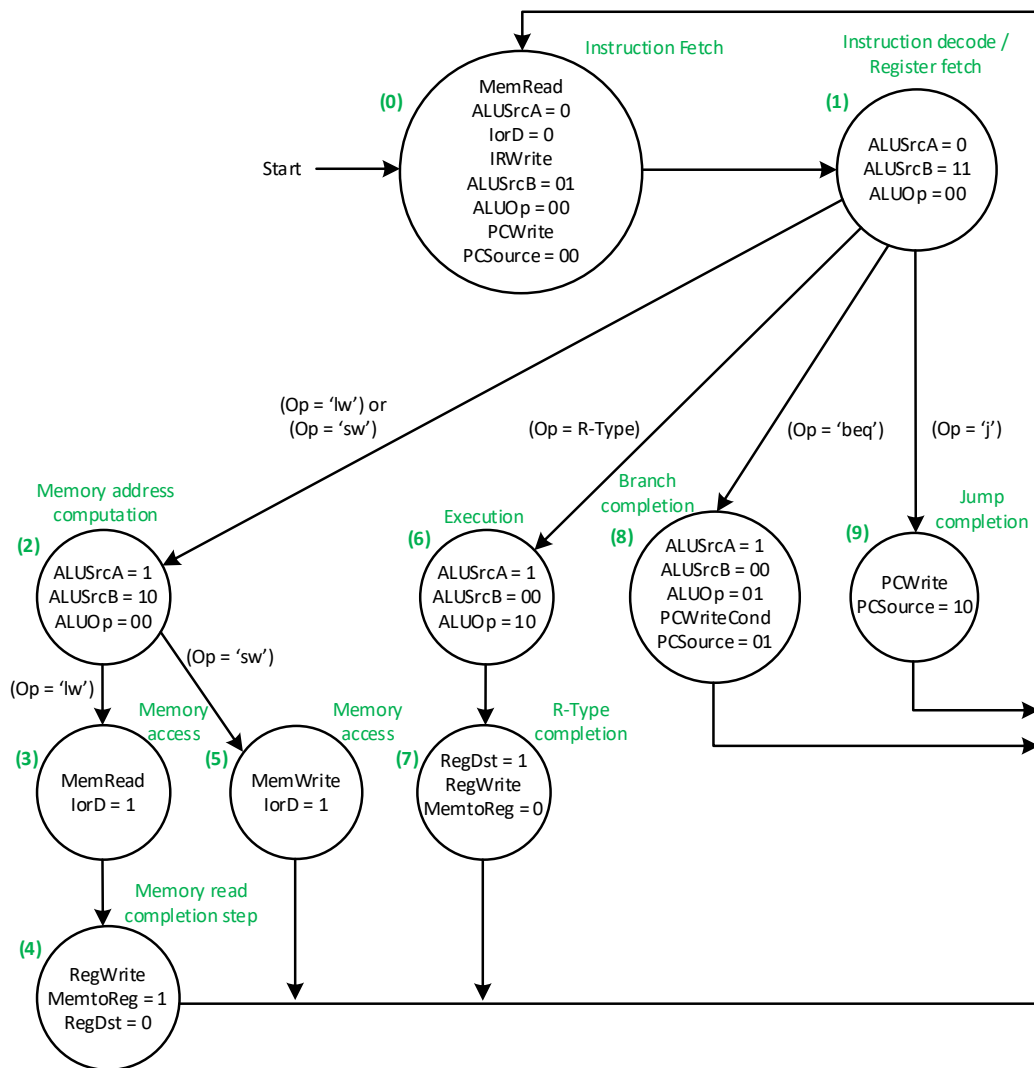
- מעבד חד-מחזורי – כל הפקודות מתבצעות במחזור שיעון אחד : CPI = 1.
- מעבד רב מחזורי – מחזור השעון תלוי בסוג הפקודה :
 - פקודת lw דורשת 5 מחזורי שיעון : CPI(lw) = 5.
 - פקודת sw דורשת 4 מחזורי שיעון : CPI(sw) = 4.
 - פקודות ALU (סוג R-Type) דורשות 4 מחזורי שיעון : CPI(R-Type) = 4.
 - פקודות קפיצה מותנית (branch) דורשות 3 מחזורי שיעון : CPI(branch) = 3.
 - פקודות קפיצה בלתי מותנית (Jump) דורשות 3 מחזורי שיעון : CPI(jump) = 3.

מימוש יחידת הבקרה:

נממש את יחידת הבקרה באמצעות מכונת מצבים סופית FSM.

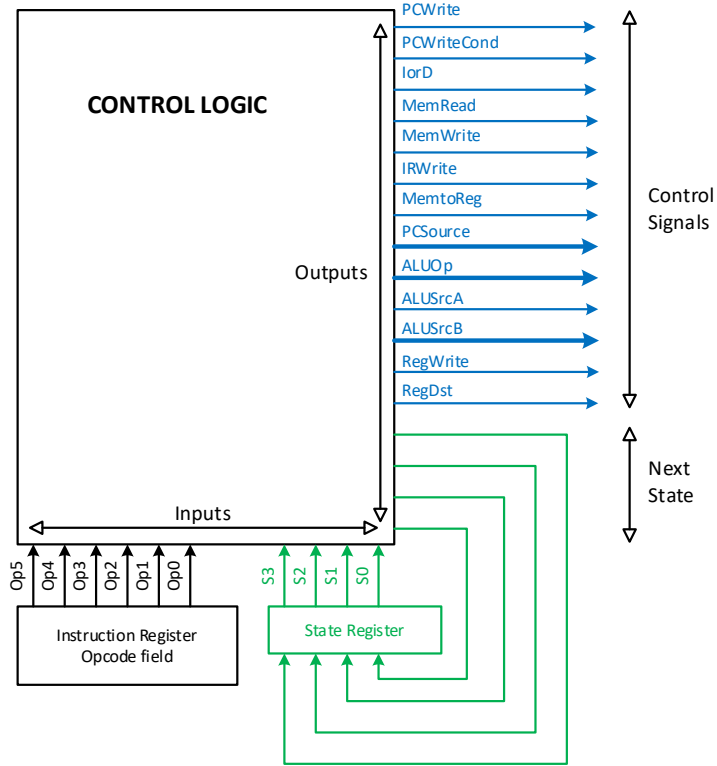
כללים בכתיבת דיאגרמות המצבים של קווי הבקרה:

- המוצאים שיצוינו בכל אחד מה-P.S. יוחזקו על ערך של 1 לוגי, בעוד ששאר המוצאים שלא יצוינו יוחזקו על ערך של 0 לוגי (ולא don't care).
- תמיד נציין את ערכי ה-Mux-ים הרלוונטיים לשלב המדובר בדיאגרמה מכיוון שיש לדעת איזה מידע עובר דרכם.
- כל מצב בדיאגרמת המצבים לוקח מחזור שעות אחד.
- המעברים ממצב למצב (קשתות) יכללו את ערך ה-Opcode שמגדיר את הפקודה.

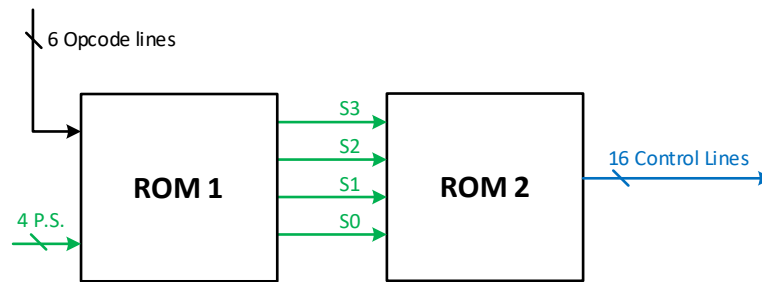


מימוש חומרתי של יחידת הבקרה ע"י יחידות ROM:

להלן מימוש יסודי:



ניתן לצמצם את המימוש ע"י שימוש בשתי יחידות ROM קטנות יותר:



מימוש חומרתי של יחידת הבקרה ע"י מערך PLA:

