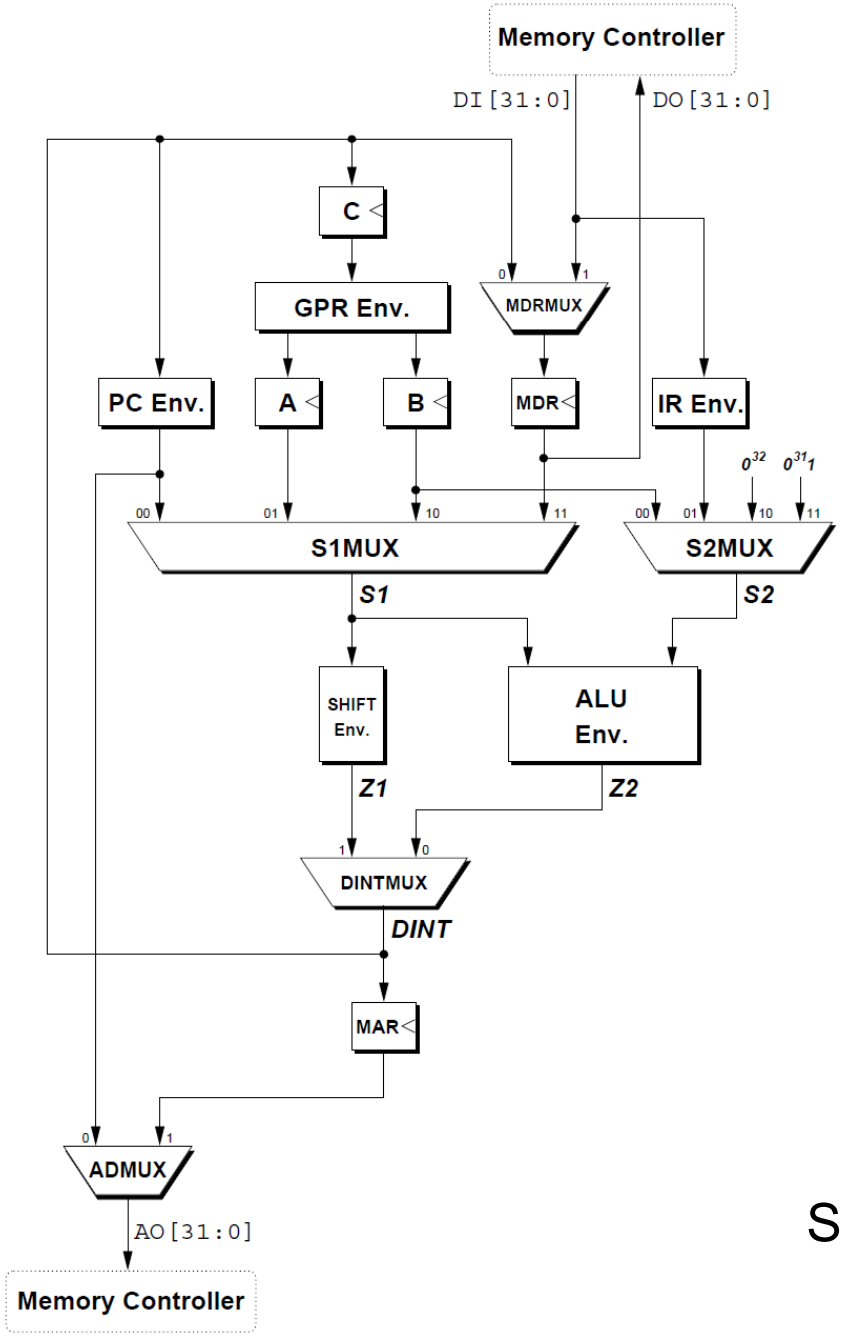



# A Simplified DLX: Implementation


**Guy Even and Moti Medina**

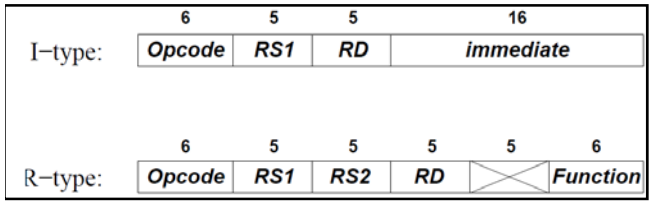
School of Electrical Engineering Tel-Aviv Univ.


January 30, 2012

| Name | RTL Instruction | Active Control Outputs |
|---|---|---|
| Fetch | $IR = M[PC]$ | MR, IRce |
| Decode | $A = RS1$,<br>$B = RS2$<br>$PC = PC + 1$ | Ace,<br>Bce,<br>S2sel[1], S2sel[0], add, PCce |
| Alu | $C = A$ op $B$ | S1sel[0], Cce, active bits in ALUF[2:0] |
| TestI | $C = (A$ rel $imm)$ | S1sel[0], S2sel[0], Cce, test, Itype,<br>active bits in ALUF[2:0] |
| AluI(add) | $C = A + imm$ | S1sel[0], S2sel[0], Cce, add, Itype |
| Shift | $C = A$ shift $sa$<br>$sa = 1, (-1)$ | S1sel[0], Cce<br>DINTsel, shift (,right) |
| Adr.Comp | $MAR = A + imm$ | S1sel[0], S2sel[0], MARce, add |
| Load | $MDR = M[MAR]$ | MDRce, ADsel, MR, MDRsel |
| Store | $M[MAR] = MDR$ | ADsel, MW |
| CopyMDR2C | $C = MDR(\gg 0)$ | S1sel[0], S1sel[1], S2sel[1], DINTsel, Cce |
| CopyGPR2MDR | $MDR = B(\ll 0)$ | S1sel[1], S2sel[1], DINTsel, MDRce |
| WBR | $RD = C$ (R-type) | GPR_WE |
| WBI | $RD = C$ (I-type) | GPR_WE, Itype |
| Branch | branch taken? | |
| Btaken | $PC = PC + imm$ | S2sel[0], add, PCce |
| JR | $PC = A$ | S1sel[0], S2sel[1], add, PCce |
| Save PC | $C = PC$ | S2sel[1], add, Cce |
| JALR | $PC = A$<br>$R31 = C$ | S1sel[0], S2sel[1], add, PCce,<br>GPR_WE, jlink |

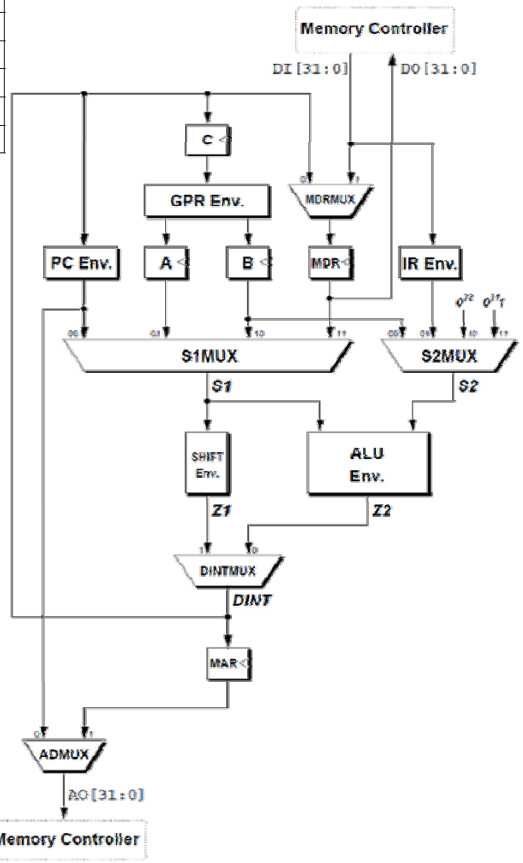
| Signal | Value | Semantics |
|---|---|---|
| ALUf[2:0] | | Controls the functionality of ALU |
| Rce | | Register clock enable |
| S1sel[1:0] | 00 | PC |
| | 01 | A |
| | 10 | B |
| | 11 | MDR |
| S2sel[1:0] | 00 | B |
| | 01 | IR |
| | 10 | 0 |
| | 11 | 1 |
| DINTsel | 0 | ALU |
| | 1 | Shifter |
| MDRsel | 0 | DINT |
| | 1 | DI |
| ADsel | 0 | PC |
| | 1 | MAR |
| shift | | explicit Shift-Instruction |
| right | | Shift to the right |
| add | | Forces an addition |
| test | | Forces a test (in the ALU) |
| MR | | Memory Read |
| MW | | Memory Write |
| GPR_WE | | GPR write enable |
| itype | | Itype-Instruction |
| jlink | | jump and link |

| condition | when does it equal 1? |
|---|---|
| D1 | special NOP |
| D2 | beqz, bnez |
| D3 | jalr |
| D4 | jr |
| D5 | lw, sw |
| D6 | sgti, seqi, sgei, slti, snei, slei |
| D7 | addi |
| D8 | sll, srl |
| D9 | add, sub, and, or, xor |
| D10 | halt |



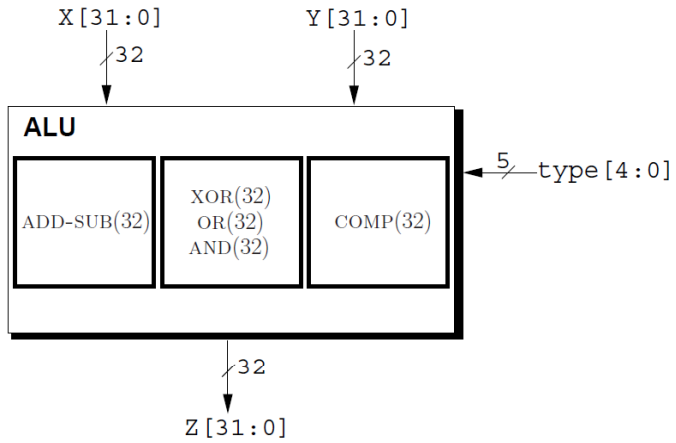






# Executing Instructions - add

| Instruction | Semantics |
|---|---|
| add RD RS1 RS2 | RD := RS1 + RS2 |




| $type[4:2]$ | $type[1]$ | $type[0]$ | $f_{type}(\vec{x},\vec{y})$ |
|---|---|---|---|
| 001 | 1 | 0 | $[\vec{x}] > [\vec{y}]$ |
| 010 | 0 | 0 | $[\vec{x}] - [\vec{y}] \pmod{2^{32}}$ |
| 010 | 1 | 0 | $[\vec{x}] = [\vec{y}]$ |
| 011 | 0 | 0 | $[\vec{x}] + [\vec{y}] \pmod{2^{32}}$ |
| 011 | 1 | 0 | $[\vec{x}] \geq [\vec{y}]$ |
| 100 | 0 | 0 | $\mathrm{XOR}(\vec{x},\vec{y})$ |
| 100 | 1 | 0 | $[\vec{x}] < [\vec{y}]$ |
| 101 | 0 | 0 | $\mathrm{OR}(\vec{x},\vec{y})$ |
| 101 | 1 | 0 | $[\vec{x}] \neq [\vec{y}]$ |
| 110 | 0 | 0 | $\mathrm{AND}(\vec{x},\vec{y})$ |
| 110 | 1 | 0 | $[\vec{x}] \leq [\vec{y}]$ |
| *** | * | 1 | $[\vec{x}] + [\vec{y}] \pmod{2^{32}}$ |

| IR[5 : 0] | Mnemonic | Semantics |
|---|---|---|
| 100 011 | add | RD = RS1 + RS2 |

# Executing Instructions – beqz, jalr

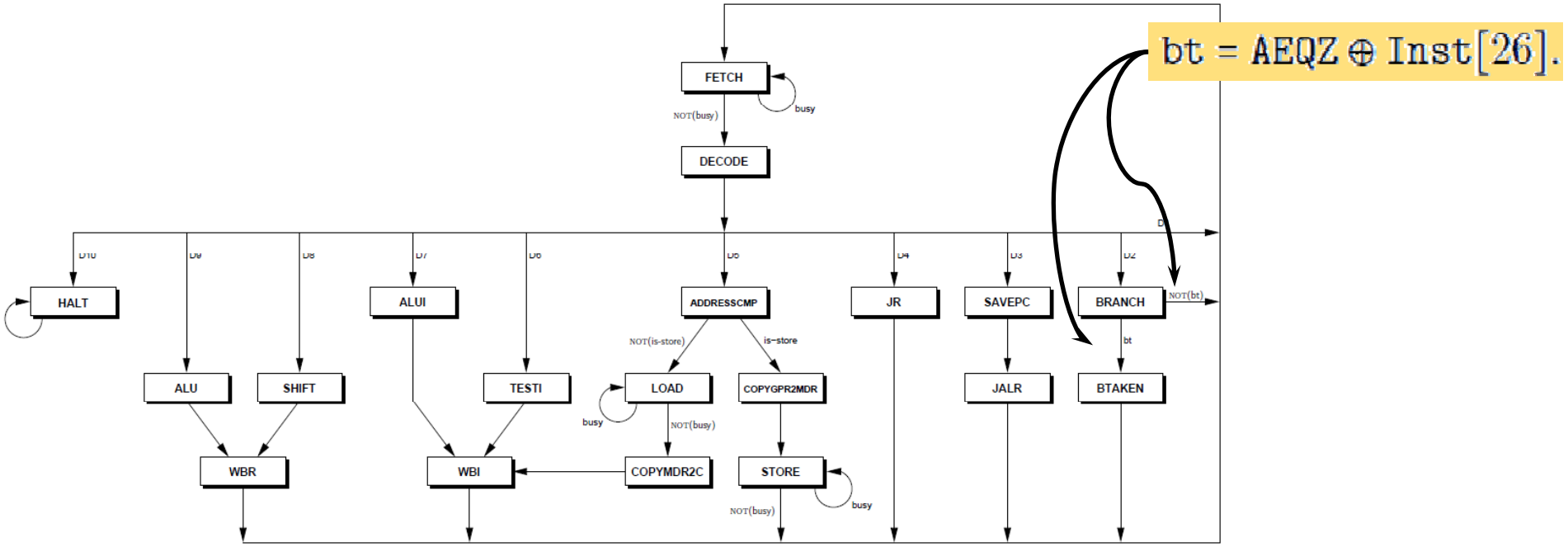
| Instruction | Semantics |
|---|---|
| beqz RS1 imm | PC = PC + 1 + sext(imm), if RS1 = 0<br>PC = PC + 1, if RS1 ≠ 0 |
| bnez RS1 imm | PC = PC + 1, if RS1 = 0<br>PC = PC + 1 + sext(imm), if RS1 ≠ 0 |
| jr RS1 | PC = RS1 |
| jalr RS1 | R31 = PC+1; PC = RS1 |

Inst[26]

000 100 || beqz

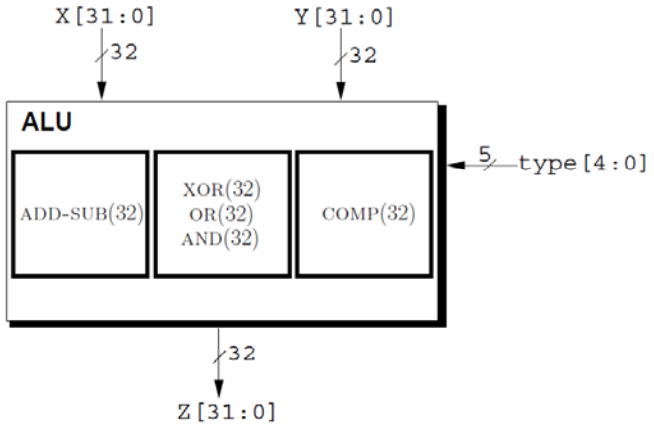
000 101 || bnez

$bt = AEQZ \oplus Inst[26].$



# Executing Instructions - slti

| Instruction | Semantics |
|---|---|
| s*rel*i RD RS1 imm | RD := 1, if condition is satisfied,<br>RD := 0 otherwise |
| if *rel* =lt | test if RS1 < sext(imm) |




| $type[4:2]$ | $type[1]$ | $type[0]$ | $f_{type}(\vec{x},\vec{y})$ |
|---|---|---|---|
| 001 | 1 | 0 | $[\vec{x}] > [\vec{y}]$ |
| 010 | 0 | 0 | $[\vec{x}] - [\vec{y}] \pmod{2^{32}}$ |
| 010 | 1 | 0 | $[\vec{x}] = [\vec{y}]$ |
| 011 | 0 | 0 | $[\vec{x}] + [\vec{y}] \pmod{2^{32}}$ |
| 011 | 1 | 0 | $[\vec{x}] \geq [\vec{y}]$ |
| 100 | 0 | 0 | $\mathrm{XOR}(\vec{x},\vec{y})$ |
| 100 | 1 | 0 | $[\vec{x}] < [\vec{y}]$ |
| 101 | 0 | 0 | $\mathrm{OR}(\vec{x},\vec{y})$ |
| 101 | 1 | 0 | $[\vec{x}] \neq [\vec{y}]$ |
| 110 | 0 | 0 | $\mathrm{AND}(\vec{x},\vec{y})$ |
| 110 | 1 | 0 | $[\vec{x}] \leq [\vec{y}]$ |
| *** | * | 1 | $[\vec{x}] + [\vec{y}] \pmod{2^{32}}$ |

| IR[31 : 26] | Mnemonic | Semantics |
|---|---|---|
| 011 100 | slti | RD = (RS1 < sext(imm)) |

# Executing Instructions - sw

| Load/Store | Semantics |
|---|---|
| lw RD RS1 imm | RD := M[sext(imm)+RS1] |
| sw RD RS1 imm | M[sext(imm)+RS1] := RD |

# ``Upgrading'' the DLX – Example 1

ברצוננו להוסיף לשפת המכונה של ה – Simplified DLX

פקודת I-type חדשה:

```
Chkbit17 RS1 RD
```

פקודה זו גורמת לעידכון RD באופן הבא:

$$RD = \begin{cases} 0^{31} \cdot 1 & if \;\; RS1[17] = 1 \\ 0^{32} & otherwise \end{cases}$$

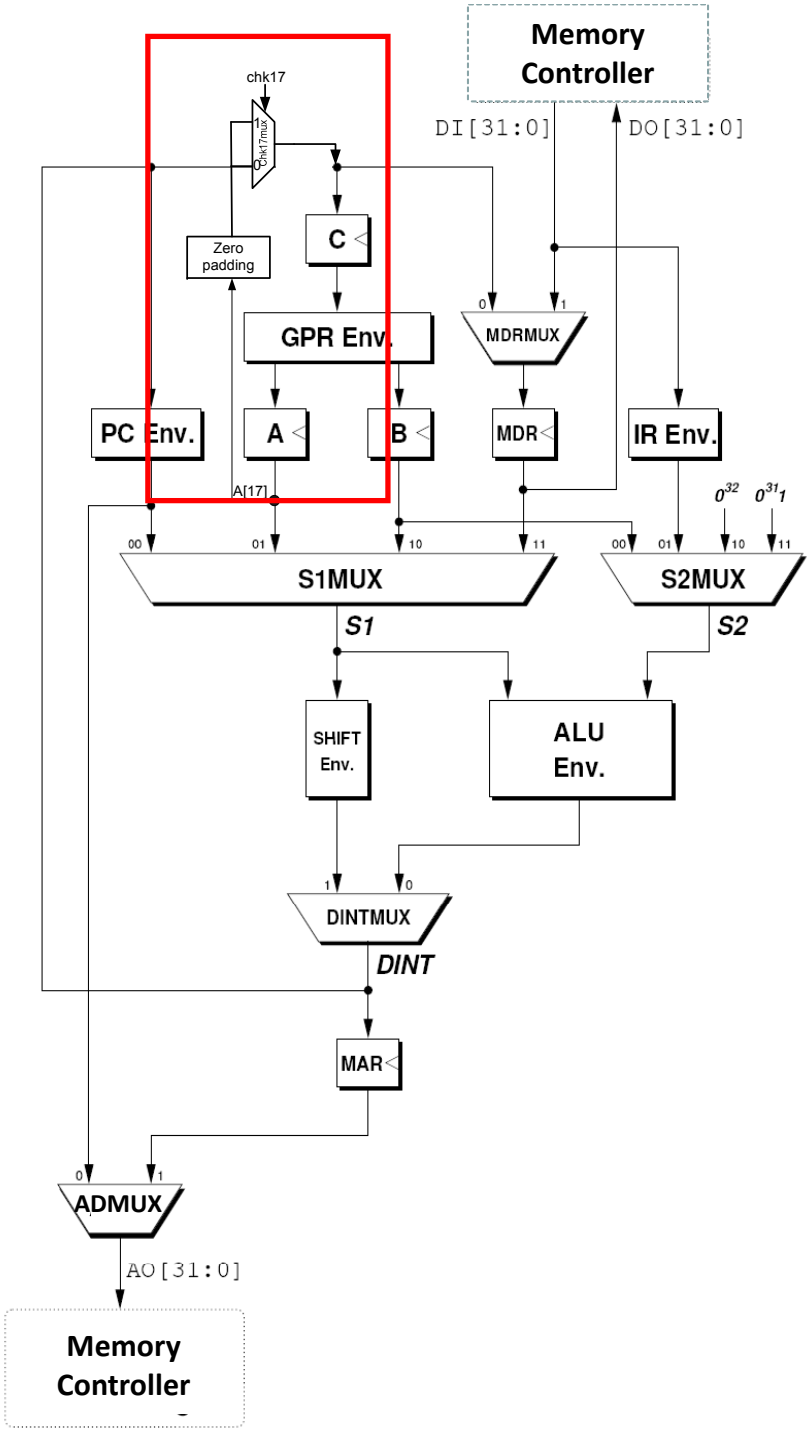
הציעו מימוש של ה-DLX שתומך בפקודה החדשה תוך ביצוע שינויים קטנים ככל האפשר במסלול הנתונים (הניקוד יופחת עבוק שינויים מוגזמים).

1. מנו את השינויים הנדרשים במסלול הנתונים על מנת לתמוך בהרצת הפקודה החדשה.
2. הציעו הרחבה לדיאגרמת המצבים של הבקרה על מנת לתמוך בהרצת הפקודה החדשה. ציירו את מסלול מצבי הבקרה, שמכונת המצבים חולפת דרכו בעת הרצת הפקודה החדשה. לכל מצב לאורך מסלול זה (חדש וישן), תארו את פקודת ה – RTL, שמתבצעת בו.

***תשובה:***

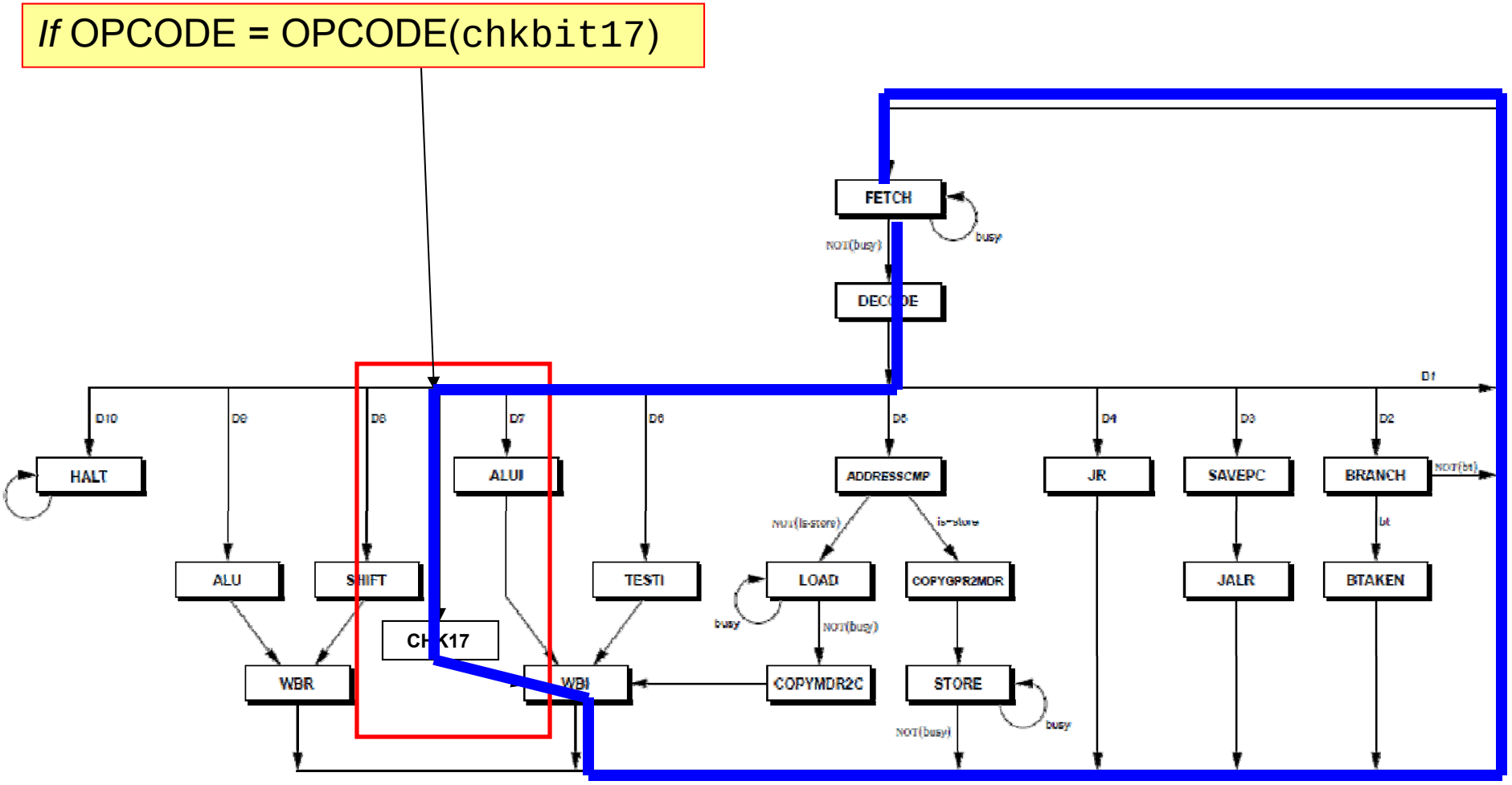
4.1) נוסיף (מסומן באדום):

- mux
- Zero Padding- משרשר 31 אפסים משמאל.
- נמשוך את הביט ה 17 מרגיסטר A.



4.2) נוסיף מצב נוסף – CHK17.

מסלול ביצוע הפקודה מסומן בכחול.



| Name | RTL Instruction | Active Control Outputs |
|---|---|---|
| Fetch | $IR = M[PC]$ | MR, IRce |
| Decode | $A = RS1$,<br>$B = RS2$<br>$PC = PC + 1$ | Ace,<br>Bce,<br>S2sel[1], S2sel[0], add, PCce |
| Alu | $C = A$ op $B$ | S1sel[0], Cce, active bits in ALUF[2:0] |
| TestI | $C = (A$ rel $imm)$ | S1sel[0], S2sel[0], Cce, test, Itype,<br>active bits in ALUF[2:0] |
| AluI(add) | $C = A + imm$ | S1sel[0], S2sel[0], Cce, add, Itype |
| Shift | $C = A$ shift $sa$<br>$sa = 1, (-1)$ | S1sel[0], Cce<br>DINTsel, shift (,right) |
| Adr.Comp | $MAR = A + imm$ | S1sel[0], S2sel[0], MARce, add |
| Load | MDR = $M$[MAR] | MDRce, ADsel, MR, MDRsel |
| Store | $M$[MAR] = MDR | ADsel, MW |
| CopyMDR2C | $C$ = MDR($\gg 0$) | S1sel[0], S1sel[1], S2sel[1], DINTsel, Cce |
| CopyGPR2MDR | MDR = $B(\ll 0)$ | S1sel[1], S2sel[1], DINTsel, MDRce |
| WBR | $RD = C$ (R-type) | GPR_WE |
| WBI | $RD = C$ (I-type) | GPR_WE, Itype |
| Branch | branch taken? | |
| Btaken | $PC = PC + imm$ | S2sel[0], add, PCce |
| JR | $PC = A$ | S1sel[0], S2sel[1], add, PCce |
| Save PC | $C = PC$ | S2sel[1], add, Cce |
| JALR | $PC = A$<br>$R31 = C$ | S1sel[0], S2sel[1], add, PCce,<br>GPR_WE, jlink |
| CHK17 | $C = 0^{31} \cdot A[17]$ | Chk17,Cce |

4.2) שימו לב שרשומים אותות הבקרה **הפעילים**, כך שאין פגיעה/שינוי במצבים אחרים.
נוסיף את מצב CHK17 (בכחול) ונסיים.

# ``Upgrading the DLX'' – Example 2

*We would like to add the following R-Type instruction:*

$$swap \quad RS1 \quad RS2$$

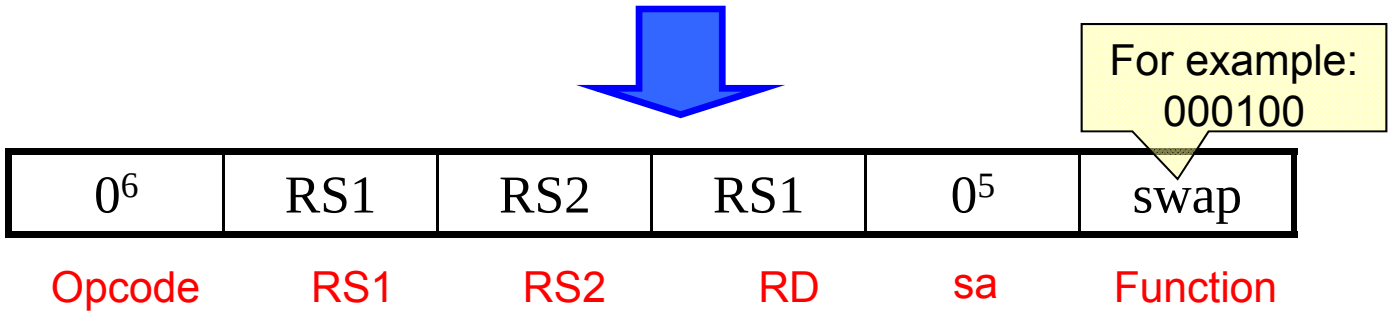
*Semantics:*

Swap between the contents of *RS1* and *RS2*.

*Advised stage:*

One should consider the instruction coding.

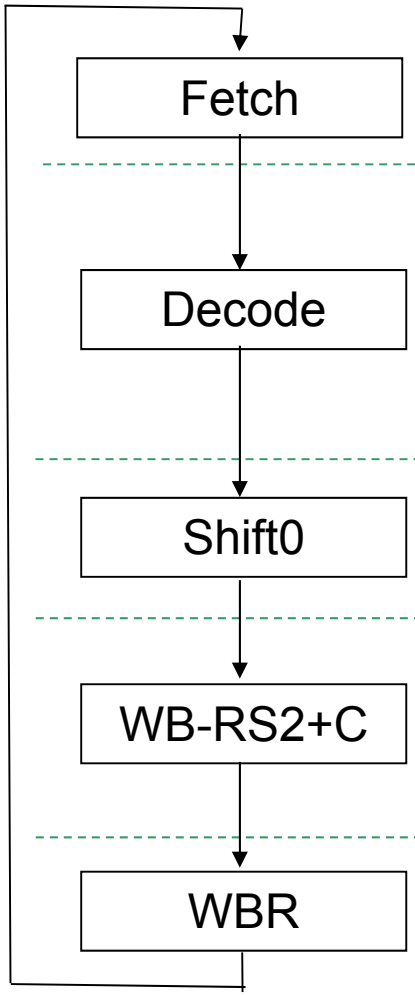
In this case: $swap \quad RS1 \quad RS2$




*Changes to be made to the Data-Path and the Control's FSM:*

There is no need to change the Data-Path.

# Control Changes & RTL instructions of each state in the instruction execution traversal.




| | RTL Instruction | Active Control Signals | |
|---|---|---|---|
| Fetch | $IR = M(PC)$ | $IRce$ , $MR$ | |
| Decode | $A = RS1$ | $Ace$ | |
| | $B = RS2$ | $Bce$ | |
| | $PC = PC + 1$ | $S2sel[1]$, $S2sel[0]$, $PCce$ , $add$ | |
| Shift0 | $C = A$ | $S1sel[0]$, $DINTsel$ , $Cce$ | NEW |
| WB-RS2+C | $RS2 = C$ | $Itype$ , $GPR\_WE$ | RD = RS2 |
| | $C = B$ | $S1sel[1]$, $DINTsel$ , $Cce$ | NEW |
| WBR | $RD = C$ | $GPR\_WE$ | RD = RS1 |

בהצלחה !!