

**HACETTEPE UNIVERSITY**

**ELECTRICAL & ELECTRONICS ENGINEERING**

**INTEGRATED CIRCUIT DESIGN (ELE419)**

**8 BIT CPU DESIGN**

SELÇUK HOLEP 21528191

BURAK CELAL KAN 21728385

# Design Specifications

* **180nm** feature size,
* **5000** transistors,
* **2** general purpose registers,
* **16** different instructions.
* Up to **50Mhz** clock speed in simulation (IRSIM).

# Design Steps

## Stepper

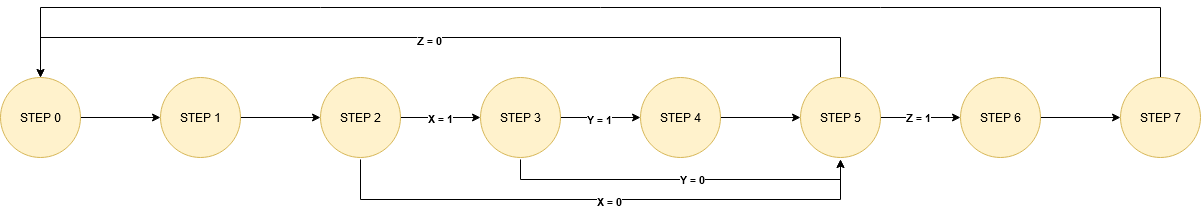
The stepper is a component which help to control unit when it will be control other components. In other words, stepper controls the control unit’s life cycle.

Totally 8 steps there are. The steps are explained below:

|  |  |
| --- | --- |
| **STEP 0** | Data on program counter(PC) to memory address register(MAR). |
| **STEP 1** | Data on ram to instruction register(IR). As the same time increase PC. |
| **STEP 2** | If there needs to be take an operand as immediate data or ram data, **repeat step 0**. If it is not necessary, go to **step 5**. |
| **STEP 3** | Data on RAM to temp register(TR)**\***.As the same time increase PC.  \****Temp register(TR)*** *is a register which stores operands (Immediate data, RAM)* |
| **STEP 4** | If the operand is RAM address of data, set this address to MAR. |
| **STEP 5** | Enable ALU or Control Unit Decoder (CUD) by checking instruction type;  ( 0 🡪 CU instruction , 1 🡪 ALU instruction).  After decoding if the instruction;   * is a CU instruction, execute and return step 0. * Is a ALU instruction, write destination data to ALU Temp Register(ATR). |
| **STEP 6** | Read source data. |
| **STEP 7** | Write value of accumulator to destination. |

The stepper component has 3 inputs. These inputs help stepper to decide which step is following.

The state diagram of the stepper is shown below:



## Instruction Set

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **MNEMONIC** | **INSTRUCTION CODE** | | | | | | | |
| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| NOP | 0 | 0 | 0 | 0 | 0 | X | 0 | X |
| MOV | 0 | 0 | 0 | 1 | D | D | S | S |
| JMP | 0 | 0 | 1 | 0 | 0 | X | 1 | 0 |
| JE | 0 | 0 | 1 | 1 | 0 | X | 1 | 0 |
| JNE | 0 | 1 | 0 | 0 | 0 | X | 1 | 0 |
| JG | 0 | 1 | 0 | 1 | 0 | X | 1 | 0 |
| JLE | 0 | 1 | 1 | 0 | 0 | X | 1 | 0 |
| HLT | 0 | 1 | 1 | 1 | 0 | X | 0 | X |
|  |  |  |  |  |  |  |  |  |
| ADD | 1 | 0 | 0 | 0 | 0 | D | S | S |
| SUB | 1 | 0 | 0 | 1 | 0 | D | S | S |
| AND | 1 | 0 | 1 | 0 | 0 | D | S | S |
| OR | 1 | 0 | 1 | 1 | 0 | D | S | S |
| NOT | 1 | 1 | 0 | 0 | 0 | D | 0 | X |
| CMP | 1 | 1 | 0 | 1 | 0 | D | S | S |
| SHL | 1 | 1 | 1 | 0 | 0 | D | 0 | X |
| SHR | 1 | 1 | 1 | 1 | 0 | D | 0 | X |

|  |  |  |
| --- | --- | --- |
| **Destination / Source** | |  |
| **D3/D1** | **D2/D0** | **DESCRIPTION** |
| 0 | X | Register |
| 1 | 0 | Immediate Data |
| 1 | 1 | Memory |

## Control Unit

**Representations:**

S1-7 🡪 Stepper’s output step,

CUD1-7 🡪 Control unit output,

FR 🡪 Flag register,

ATR 🡪 ALU temp register,

ACC 🡪 Accumulator,

SRC 🡪 Source operand decoder,

DST 🡪 Destination operand decoder.

**Logic Functions:**

|  |  |
| --- | --- |
| PCUP | S1 + S3 |
| PCR | S0 + S2 |
| PCW | CUD2 + CUD3.FRE + CUD4.(FRE)' + CUD5.FRG + CUD6.(FRL+FRE) |
| MARW | S0 + S2 + S4 |
| IRW | S1 |
| SRCE | S5 + S6 + S7 |
| DSTE | S5 + S6 + S7 |
| CUDE | S5.(IR7)' |
| SRESET | RST + (IR7)'.IR6.IR5.IR4 |
| Sx | IR1 + IR3 |
| Sy | IR0.IR1 + IR2.IR3 |
| Sz | IR7 |
| ALUE | IR7 |
| ATRW | S5.IR7 |
| ACCR | S7.IR7 |
| FRCLK | S6.IR7 |
| TRW | S3 |
| TRR | S4 + SRC2.(CUD1 + CUD2 + CUD3 + CUD4 + S6.IR7) |
| R1R | SRC0.(CUD1 + S6.IR7) + DST0.IR7.S5 |
| R1W | DST0.(CUD1 + S7.IR7.((IR6)' + IR5 + (IR4)')) |
| R2R | SRC1.(CUD1 + S6.IR7) + DST1.IR7.S5 |
| R2W | DST1.(CUD1 + S7.IR7.((IR6)' + IR5 + (IR4)')) |
| RAMR | S1 + S3 + SRC3.(CUD1 + S6.IR7) + DST3.IR7.S5 |
| RAMW | DST3.(CUD1 + S7.IR7.((IR6)' + IR5 + (IR4)')) |

## Simulation Results

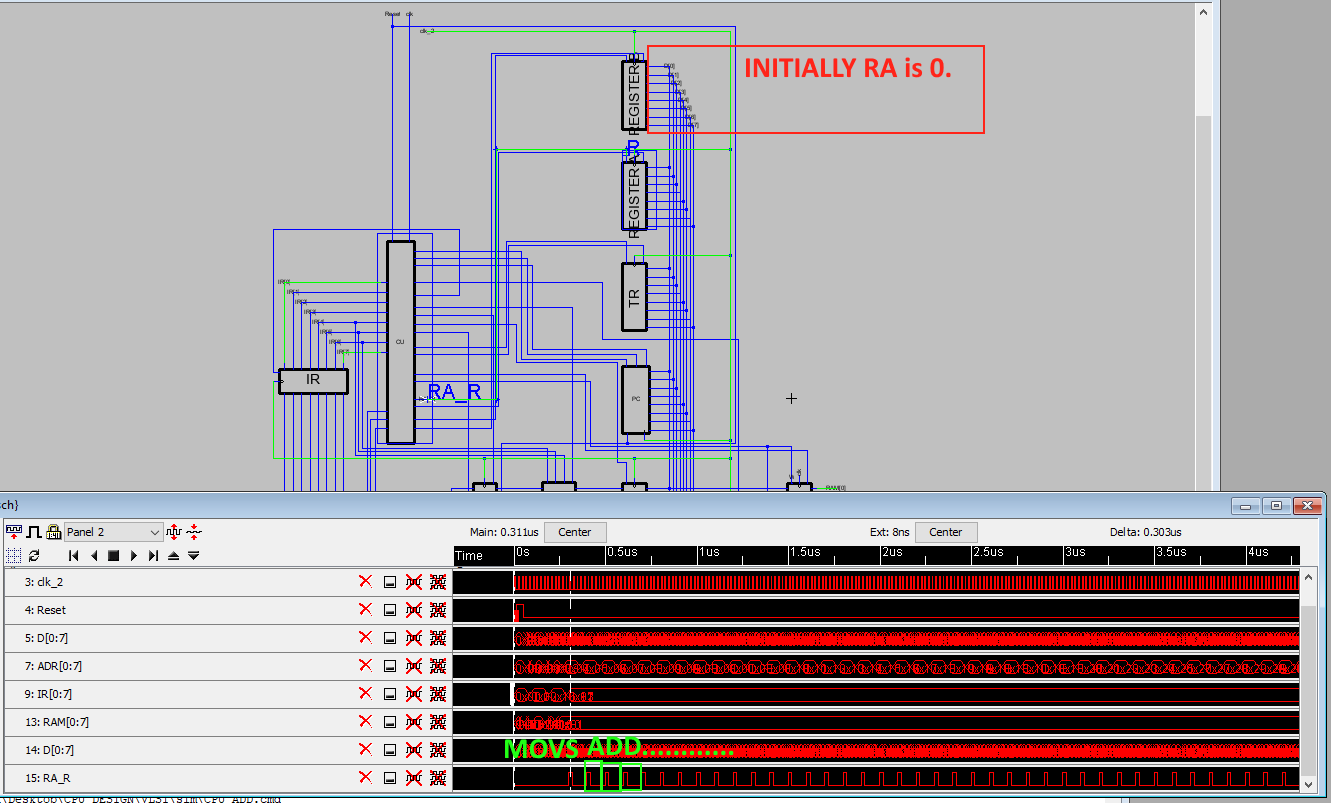
The simulations are made by using IRSIM.

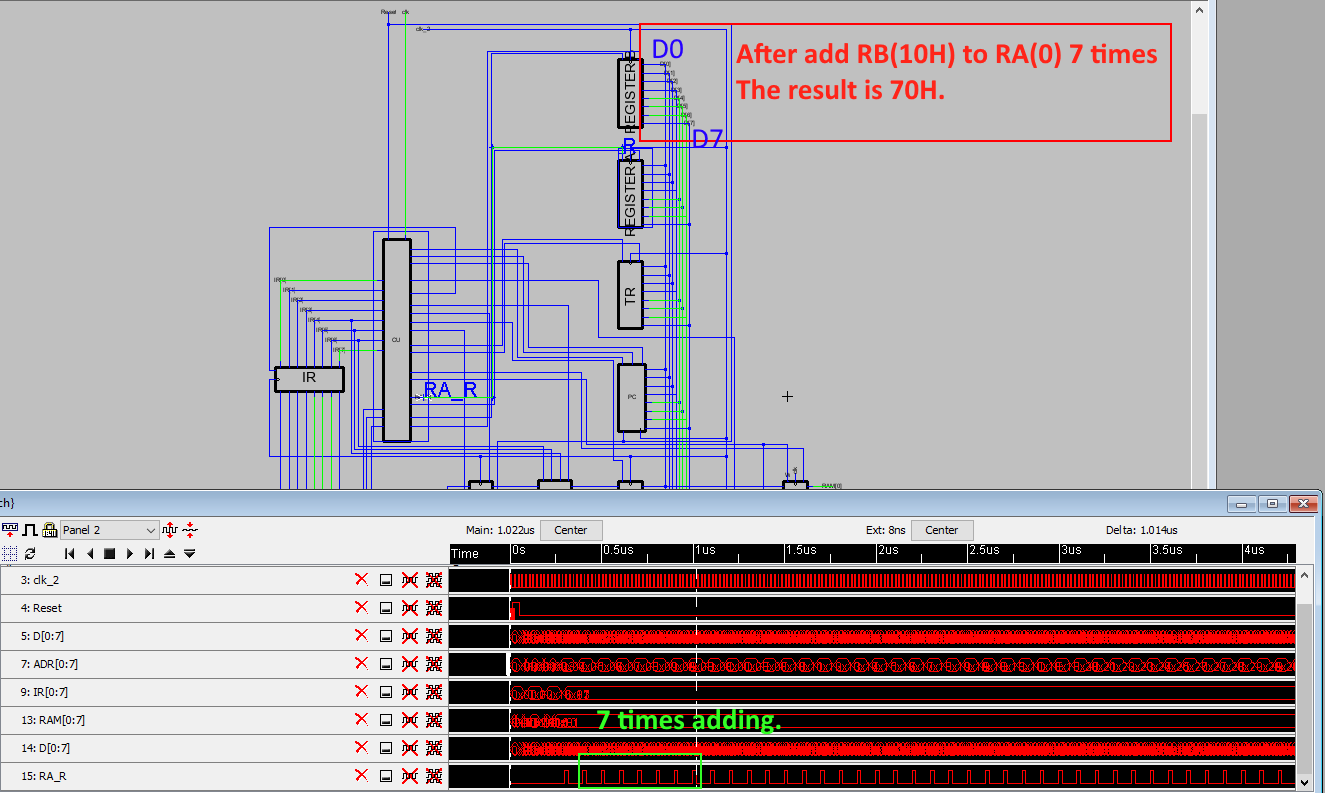
**ADD:** This exampleadds RB to RA infinitely.

MOV RA, 0x10

MOV RB, 0x07

ADD RA, RB





**CMP + JLE:**

MOV RA, 0xFE

MOV RB, 0xFF

CMP RA, RB

JLE 0x52

