|  |
| --- |
|  |
| Architecture Project |
| Simple PDP-11 |
|  |
| **Team 15** |
|  |

Team Members Sec Bn

## **-Evram Youssef 1 9**

## **-Remonda Talaat 1 20**

## **-Mohamed Shawky 2 16**

## **-Mahmoud Othman 2 21**

Submitted to: TA.Ali el-Seddiq

Under supervision of: Dr. Hwaida

* Two operands instructions:
* Syntax “Opcode Src, Dst”

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction Name** | **Op Code (Binary)** | **Operation Performed** | **Condition codes** |
| Mov | 0001 | Dst ← [Src] | “N” set if [src] < 0  “Z” set if [src] = 0  “V” reset |
| ADD | 0010 | Dst ← [Dst] + [Src] | “N” set if result < 0  “Z” set if result = 0  “V” set if arithmetic overflow occurs  “C” set if carry from the MSB of result occurs |
| ADC | 0011 | Dst ← [Dst] + [Src] + C | “N” set if result < 0  “Z” set if result = 0  “V” set if arithmetic overflow occurs  “C” set if carry from the MSB of result occurs |
| SUB | 0100 | Dst ← [Dst] – [Src] | “N” set if result < 0  “Z” set if result = 0  “V” set if arithmetic overflow occurs  “C” set if no carry from the MSB of result occurs |
| SBC | 0101 | Dst ← [Dst] – [Src] – C | “N” set if result < 0  “Z” set if result = 0  “V” set if arithmetic overflow occurs  “C” set if no carry from the MSB of result occurs |
| AND | 0110 | Dst ← [Dst] AND [Src] | “N” set if MSB of result = 0  “Z” set if [src] = 0  “V” reset |
| OR | 0111 | Dst ← [Dst] OR [Src] | “N” set if MSB of result = 0  “Z” set if [src] = 0  “V” reset |
| XNOR | 1000 | Dst ← [Dst] XNOR [Src] | “N” set if MSB of result = 0  “Z” set if [src] = 0  “V” reset |
| CMP | 1001 | [Dst] – [Src]  Neither of the operands are affected | “N” set if result < 0  “Z” set if result = 0  “V” set if arithmetic overflow occurs  “C” set if no carry from the MSB of result occurs |

* One operand instructions:
* Syntax “Opcode Dst”

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction Name** | **Op Code (Binary)** | **Operation Performed** | **Condition codes** |
| INC | 11110000 | Dst ← [Dst] + 1 | “N” set if result < 0  “Z” set if result = 0  “V” set if [Dst] was 077777 |
| DEC | 11110001 | Dst ← [Dst] - 1 | “N” set if result < 0  “Z” set if result = 0  “V” set if [Dst] was 100000 |
| CLR | 11110010 | Dst ← 0 | “N” reset  “V” reset  “C” reset  “Z” set |
| INV | 11110011 | Dst ← INV([Dst]) | “N” set if MSB of result = 1  “V” reset  “C” set  “Z” set if result = 0 |
| LSR | 11110100 | Dst ← 0 || [Dst]15->1 | “N” set if MSB of result = 1  “V” = N XOR C  “C” is loaded by last bit shifted out of register  “Z” set if result = 0 |
| ROR | 11110101 | Dst ← [Dst]0 || [Dst]15->1 | “N” set if MSB of result = 1  “V” = N XOR C  “C” is loaded by last bit shifted out of register  “Z” set if result = 0 |
| RRC | 11110110 | Dst ← C || [Dst]15->1 | “N” set if MSB of result = 1  “V” = N XOR C  “C” is loaded by last bit shifted out of register  “Z” set if result = 0 |
| ASR | 11110111 | Dst ← [Dst]15 || [Dst]15->1 | “N” set if MSB of result = 1  “V” = N XOR C  “C” is loaded by last bit shifted out of register  “Z” set if result = 0 |
| LSL | 11111000 | Dst ← [Dst]14->0 || 0 | “N” set if MSB of result = 1  “V” = N XOR C  “C” is loaded by last bit shifted out of register  “Z” set if result = 0 |
| ROL | 11111001 | Dst ← [Dst]14->0 || [Dst]15 | “N” set if MSB of result = 1  “V” = N XOR C  “C” is loaded by last bit shifted out of register  “Z” set if result = 0 |
| RLC | 11111010 | Dst ← [Dst]14->0 || C | “N” set if MSB of result = 1  “V” = N XOR C  “C” is loaded by last bit shifted out of register  “Z” set if result = 0 |

* Branch instructions:
* Syntax “Opcode Offset”
* Operation “PC ← PC + Offset” is performed if branch condition is TRUE

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction Name** | **Op Code (Binary)** | **Operation Condition** | **Condition codes** |
| BR | 000000 | No Condition | \_\_\_\_ |
| BEQ | 000001 | Z = 1 | \_\_\_\_ |
| BNE | 000010 | Z = 0 | \_\_\_\_ |
| BLO | 000011 | C = 0 | \_\_\_\_ |
| BLS | 110000 | C = 0 OR Z = 1 | \_\_\_\_ |
| BHI | 110001 | C = 1 | \_\_\_\_ |
| BHS | 110011 | C = 1 OR Z = 1 | \_\_\_\_ |

* No operation instructions:
* Syntax “Opcode”

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction Name** | **Op Code (Binary)** | **Operation Performed** | **Condition codes** |
| HLT | 1010 | Stop the processor | \_\_\_\_ |
| NOP | 1011 | No operation is performed  Continue code | \_\_\_\_ |

* Jump Subroutine instructions:

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction Name** | **Op Code (Binary)** | **Syntax** | **Condition codes** |
| JSR | 1101 | JSR Address |  |
| RTS | 111001 | RTS |  |
| INTERRUPT | 111010 | \_\_\_ |  |
| IRET | 111011 | IRET |  |

* ALU Operations:
* ADD : F = Temp0 + Busout.
* SUB: F = Busout - Temp0.
* AND: F = Temp0 AND Busout.
* OR: F = Temp0 OR Busout.
* NOT: F = NOT Busout.
* XNOR: F = Temp0 XNOR Busout.
* LSR: F = LSR Busout.
* ROR: F = ROR Busout.
* RRC: F = RRC Busout.
* ASR: F = ASR Busout.
* LSL: F = LSL Busout.
* ROL: F = ROL Busout.
* RLC: F = ROL Busout.
* DEC: F = Busout - 1.
* INC: F = Busout + 1.
* OUTZero: F = 0.
* Control Store Groups:
* Group 0 : “6-Bits”
  + Next Address Field.
* Group 1(out group): “3-Bits”
  + No Transfer.
  + PCout.
  + IRout.
  + MDRout.
  + Rsrcout.
  + Rdestout.
  + Temp1out.
  + ALUout.
* Group 2 (in group): “3-Bits ”
  + No Transfer.
  + PCin.
  + IRin.
  + Rsrcin.
  + Rdestin.
  + FLAGSin.
* Group 3(in group): “2-Bits”
  + No Transfer.
  + MARin.
  + MDRin.
  + Temp1in.
* Group 4: “2-Bits”
  + No Action.
  + Temp0in.
  + Clear Temp0.
* Group 5: “3-Bits”
  + No operation.
  + ALUadd.
  + ALUbus+1.
  + ALUbus-1.
  + ALUop.
* Group 6: “2-Bits”
  + Read.
  + Write.
  + No Action.
* Group 7: “1-Bit”
  + Carry in = 1.
  + Carry in = 0.
* Group 8: “1-Bit”
* No Action
* PLAout.
* Group 9 (PLA Selectors): “3-Bits”
  + No Action.
  + ORdest
  + ORdestindirect.
  + ORsrcindirect.
  + ORresult.
  + ORop.
  + Address\_fieldout.
  + Flagsout.
* Flag Register:
* Bit 0: Carry Flag.
* Bit 1: Zero Flag.
* Bit 2: Negative Flag.
* Bit 3: Parity Flag.
* Bit 4: Overflow Flag.
* Addressing Modes:
* Register Direct: 000
* Auto Increment: 001
* Auto Decrement: 010
* Indexed: 011
* Register Indirect: 100
* Auto Increment Indirect: 101
* Auto Decrement Indirect: 110
* Indexed Indirect: 111
* Control Store:

|  |  |  |
| --- | --- | --- |
| Address | Control Word | Next Address Field |
| 000000 | PCout , MARin , Read , ALUbus+1 | 000001 |
| 000001 | ALUout, PCin | 000010 |
| 000010 | MDRout, IRin, uAR ← [PLA] | 000011 |
| 000011 | Rsrcout, Temp1in, uAR ← [ORdest] | 010000 |
| 000100 | Rsrcout, MARin, Read, uAR ← [ORsrc.ind] | 001110 |
| 000101 | Rsrcout, MARin , Read , ALUbus+1 | 000110 |
| 000110 | ALUout, Rsrc.in, uAR ← [ORsrc.ind] | 001110 |
| 000111 | Rsrcout, ALUbus-1 | 001000 |
| 001000 | ALUout, Rsrc.in, MARin, Read, uAR ← [ORsrc.ind] | 001110 |
| 001001 | PCout , MARin , Read , ALUbus+1 | 001010 |
| 001010 | ALUout, PCin | 001011 |
| 001011 | MDRout, Temp0in | 001100 |
| 001100 | Rsrcout, ALUadd | 001101 |
| 001101 | ALUout, MARin, Read, uAR ← [ORsrc.ind] | 001110 |
| 001110 | MDRout, MARin, Read | 001111 |
| 001111 | MDRout, Temp1in, uAR ← [ORdest] | 010000 |
| 010000 | Rdest.out, Temp0in, uAR ← [ORop] | 101110 |
| 010001 | Rdest.out, MARin, Read, uAR ← [ORdest.ind] | 011011 |
| 010010 | Rdest.out, MARin , Read , ALUbus+1 | 010011 |
| 010011 | ALUout, Rdest.in, uAR ← [ORdest.ind] | 011011 |
| 010100 | Rdest.out, ALUbus-1 | 010101 |
| 010101 | ALUout, Rdest.in, MARin, Read, uAR ← [ORdest.ind] | 011011 |
| 010110 | PCout , MARin , Read , ALUbus+1 | 010111 |
| 010111 | ALUout, PCin | 011000 |
| 011000 | MDRout, Temp0in | 011001 |
| 011001 | Rdest.out, ALUadd | 011010 |
| 011010 | ALUout, MARin, Read, uAR ← [ORdest.ind] | 011011 |
| 011011 | MDRout, MARin, Read | 011100 |
| 011100 | MDRout, Temp0in, uAR ← [ORop] | 101110 |
| 011101 | Rsrcout, ALUbus-1 | 011110 |
| 011110 | ALUout, MARin, Rsrc.in | 011111 |
| 011111 | Flagsout, MDRin, WR | 100000 |
| 100000 | Rsrcout, ALUbus-1 | 100001 |
| 100001 | ALUout, MARin, Rsrc.in | 100010 |
| 100010 | PCout , MDRin , WR, uAR ← [ORop] | 101110 |
| 100011 | Address\_fieldout, PCin | 000000 |
| 100100 | Rsrcout, MARin, Read, ALUbus+1 | 100101 |
| 100101 | ALUout, Rsrc.in | 100110 |
| 100110 | MDRout , PCin, uAR ← [ORop] | 101110 |
| 100111 | Rsrcout, MARin, Read, ALUbus+1 | 101000 |
| 101000 | ALUout, Rsrc.in | 101001 |
| 101001 | MDRout , FLAGSin | 000000 |
| 101010 | uAR ← [ORresult] | 101011 |
| 101011 | PCout, Temp0in | 101100 |
| 101100 | Address\_fieldout, ALUadd | 101101 |
| 101101 | ALUout, PCin | 000000 |
| 101110 | Temp1out, MDRin, WR | 000000 |
| 101111 | Temp1out, Rdest.in | 000000 |
| 110000 | Temp1out, ALUop, uAR ← [ORdest.ind] | 110001 |
| 110001 | ALUout, Rdest.in | 000000 |
| 110010 | ALUout, MDRin, WR | 000000 |

* CPU Cycles Analysis:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Clock Cycles** | | |
| **Instruction** | **Min** | **Max** | **Avg** |
| MOV | 6 | 21 | 11 |
| ADD | 7 | 23 | 14 |
| ADC | 7 | 23 | 14 |
| SUB | 10 | 26 | 17 |
| SBC | 10 | 26 | 17 |
| AND  OR  XNOR | 7 | 23 | 14 |
|
|
| CMP | 9 | 25 | 16 |
| INC  DEC  INV | 7 | 15 | 10 |
|
|
| CLR | 6 | 12 | 8 |
| LSR  ROR  RRC  ASR  LSL  ROL  RLC | 7 | 14 | 9 |
|
|
|
|
|
|
| BR | 8 | | |
| BEQ  BHI  BHS  BLO  BLS  BNE | 9 | | |
| HLT  INT  IRET  JSR X(R)  NOP  RTS | 4 | | |

Memory Access Analysis

*Group 1: Two Operands Regular Instructions (ADD, ADC, SUB, SUBC, AND, OR, XNOR):*Addressing modes are (RGS: register, IND\_RGS: indirect register)

|  |  |  |
| --- | --- | --- |
| SRC | DST | MA |
| RGS | RGS | 1 |
| RGS | IND\_RGS | 3 |
| RGS | Autoincremet | 3 |
| RGS | IND\_autoincremetn | 4 |
| RGS | Autodecrement | 3 |
| RGS | IND\_Autodec | 4 |
| RGS | Indexed | 4 |
| RGS | Ind\_Indexed | 5 |

|  |  |  |
| --- | --- | --- |
| SRC | DST | MA |
| IND\_RGS | RGS | 2 |
| IND\_RGS | IND\_RGS | 4 |
| IND\_RGS | Autoincremet | 4 |
| IND\_RGS | IND\_autoincremetn | 5 |
| IND\_RGS | Autodecrement | 4 |
| IND\_RGS | IND\_Autodec | 5 |
| IND\_RGS | Indexed | 5 |
| IND\_RGS | Ind\_Indexed | 6 |

|  |  |  |
| --- | --- | --- |
| INC | RGS | 2 |
| INC | IND\_RGS | 4 |
| INC | Autoincremet | 4 |
| INC | IND\_autoincremetn | 5 |
| INC | Autodecrement | 4 |
| INC | IND\_Autodec | 5 |
| INC | Indexed | 5 |
| INC | Ind\_Indexed | 6 |

|  |  |  |
| --- | --- | --- |
| IND\_INC | RGS | 3 |
| IND\_INC | IND\_RGS | 5 |
| IND\_INC | Autoincremet | 5 |
| IND\_INC | IND\_autoincremetn | 6 |
| IND\_INC | Autodecrement | 5 |
| IND\_INC | IND\_Autodec | 6 |
| IND\_INC | Indexed | 6 |
| IND\_INC | Ind\_Indexed | 7 |

|  |  |  |
| --- | --- | --- |
| DEC | RGS | 2 |
| DEC | IND\_RGS | 4 |
| DEC | Autoincremet | 4 |
| DEC | IND\_autoincremetn | 5 |
| DEC | Autodecrement | 4 |
| DEC | IND\_Autodec | 5 |
| DEC | Indexed | 5 |
| DEC | Ind\_Indexed | 6 |

|  |  |  |
| --- | --- | --- |
| IND\_DEC | RGS | 3 |
| IND\_DEC | IND\_RGS | 5 |
| IND\_DEC | Autoincremet | 5 |
| IND\_DEC | IND\_autoincremetn | 6 |
| IND\_DEC | Autodecrement | 5 |
| IND\_DEC | IND\_Autodec | 6 |
| IND\_DEC | Indexed | 6 |
| IND\_DEC | Ind\_Indexed | 7 |
| IND\_INDEX | RGS | 4 |
| IND\_INDEX | IND\_RGS | 6 |
| IND\_ INDEX | Autoincremet | 6 |
| IND\_ INDEX | IND\_autoincremetn | 7 |
| IND\_ INDEX | Autodecrement | 6 |
| IND\_ INDEX | IND\_Autodec | 7 |
| IND\_ INDEX | Indexed | 7 |
| IND\_ INDEX | Ind\_Indexed | 8 |

|  |  |  |
| --- | --- | --- |
| INDEX | RGS | 3 |
| INDEX | IND\_RGS | 5 |
| INDEX | Autoincremet | 5 |
| INDEX | IND\_autoincremetn | 6 |
| INDEX | Autodecrement | 5 |
| INDEX | IND\_Autodec | 6 |
| INDEX | Indexed | 6 |
| INDEX | Ind\_Indexed | 7 |

**Group 2: One Operand Instructions (INC, DEC, CLR, INV, LSR, ROR, RRC, ASR, LSL, ROL, RLC):**

|  |  |
| --- | --- |
| **Destination** | **MA** |
| Register | **1** |
| Indirect Register | 3 |
| Autoincremet | 3 |
| Indirect Autoincremetn | 4 |
| Autodecrement | 3 |
| Indirect Autodecrement | 4 |
| Indexed | 4 |
| Indirect Indexed | 5 |

**Group 3: Two Operands Special Instructions (MOV, CMP):**As MOV doesn’t require the final data of the destination, just the address, and CMP doesn’t store the final value of the operation.

So, the second table would differ from the previous one, as MA would be less by 1, except when RGS is at the destination.

|  |  |  |
| --- | --- | --- |
| SRC | DST | MA |
| RGS | RGS | **1** |
| RGS | IND\_RGS | 2 |
| RGS | Autoincremet | 2 |
| RGS | IND\_autoincremetn | 3 |
| RGS | Autodecrement | 2 |
| RGS | IND\_Autodec | 3 |
| RGS | Indexed | 3 |
| RGS | Ind\_Indexed | 4 |

|  |  |  |
| --- | --- | --- |
| SRC | DST | MA |
| IND\_RGS | RGS | **2** |
| IND\_RGS | IND\_RGS | 3 |
| IND\_RGS | Autoincremet | 3 |
| IND\_RGS | IND\_autoincremetn | 4 |
| IND\_RGS | Autodecrement | 3 |
| IND\_RGS | IND\_Autodec | 4 |
| IND\_RGS | Indexed | 4 |
| IND\_RGS | Ind\_Indexed | 5 |

|  |  |  |
| --- | --- | --- |
| IND\_INC | RGS | **3** |
| IND\_INC | IND\_RGS | 4 |
| IND\_INC | Autoincremet | 4 |
| IND\_INC | IND\_autoincremetn | 5 |
| IND\_INC | Autodecrement | 4 |
| IND\_INC | IND\_Autodec | 5 |
| IND\_INC | Indexed | 5 |
| IND\_INC | Ind\_Indexed | 6 |

|  |  |  |
| --- | --- | --- |
| INC | RGS | **2** |
| INC | IND\_RGS | 3 |
| INC | Autoincremet | 3 |
| INC | IND\_autoincremetn | 4 |
| INC | Autodecrement | 3 |
| INC | IND\_Autodec | 4 |
| INC | Indexed | 4 |
| INC | Ind\_Indexed | 5 |

|  |  |  |
| --- | --- | --- |
| DEC | RGS | **2** |
| DEC | IND\_RGS | 3 |
| DEC | Autoincremet | 3 |
| DEC | IND\_autoincremetn | 4 |
| DEC | Autodecrement | 3 |
| DEC | IND\_Autodec | 4 |
| DEC | Indexed | 4 |
| DEC | Ind\_Indexed | 5 |

|  |  |  |
| --- | --- | --- |
| IND\_DEC | RGS | **3** |
| IND\_DEC | IND\_RGS | 4 |
| IND\_DEC | Autoincremet | 4 |
| IND\_DEC | IND\_autoincremetn | 5 |
| IND\_DEC | Autodecrement | 4 |
| IND\_DEC | IND\_Autodec | 5 |
| IND\_DEC | Indexed | 5 |
| IND\_DEC | Ind\_Indexed | 6 |
| IND\_INDEX | RGS | **4** |
| IND\_INDEX | IND\_RGS | 5 |
| IND\_ INDEX | Autoincremet | 5 |
| IND\_ INDEX | IND\_autoincremetn | 6 |
| IND\_ INDEX | Autodecrement | 5 |
| IND\_ INDEX | IND\_Autodec | 6 |
| IND\_ INDEX | Indexed | 6 |
| IND\_ INDEX | Ind\_Indexed | 7 |

|  |  |  |
| --- | --- | --- |
| INDEX | RGS | **3** |
| INDEX | IND\_RGS | 4 |
| INDEX | Autoincremet | 4 |
| INDEX | IND\_autoincremetn | 5 |
| INDEX | Autodecrement | 4 |
| INDEX | IND\_Autodec | 5 |
| INDEX | Indexed | 5 |
| INDEX | Ind\_Indexed | 6 |

**Group 4: Jumpers and Stackers (JSR, RTS, ITR, IRET):**

|  |  |
| --- | --- |
| **Operation** | **Memory Access** |
| **JSR** | **3** |
| **RTS** | **2** |
| **ITR** | **3** |
| **IRET** | **3** |

**Group 5: Branches, HLT and NO Operation:**

**Memory Access = 1 for fetching any of these instructions.**