Proiect FIC – Instruction Set

Instructions that transfer the value from AC in the registers X or Y:

|  |  |  |
| --- | --- | --- |
| Opcode | Instruction Name | Other <10> |
| 000000 | TRX | 0000000000000000 |
| 000001 | TRY | 0000000000000000 |

TRX – transfers the value from ACC into X

TRY – transfers the value from ACC into Y

The other bits are all 0 as they are unused.

Memory instructions:

Register Address: 0 – X

1 - Y

Immediate is on 9 bits, so Data Memory needs to have maximum 512 locations, each location has 16 bits.

Load and Store instructions:

|  |  |  |  |
| --- | --- | --- | --- |
| Opcode | Instruction Name | Register Address (bits) | Immediate (bits) |
| 000010 | LDR | <1> | <9> |
| 000011 | STR | <1> | <9> |

Stack instructions:

|  |  |  |  |
| --- | --- | --- | --- |
| Opcode | Instruction Name | Register Address (bits) | Immediate (bits) |
| 000100 | PSH | <2> | 00000000 |
| 000101 | POP | <2> | 00000000 |

Stack instructions can push/pop values from the following registers onto the stack:

00 – X

01 – Y

10 – ACC

11 - PC

Branch instructions:

Address is on 10 bits, so Instruction Memory needs to have maximum 1024 locations, each location has 16 bits.

|  |  |  |
| --- | --- | --- |
| Opcode | Instruction Name | Address |
| 000110 | BRZ | <10> |
| 000111 | BRN | <10> |
| 001000 | BRC | <10> |
| 001001 | BRO | <10> |
| 001010 | BRA | <10> |
|  | JMP | <10> |
|  | RET | 0000000000 |

JMP and RET will be implemented as pseudoinstructions. A JMP [adr] should generate:

PSH PC

BRA [adr]

A RET instruction should generate POP PC

Arithmetic and logic instructions:

|  |  |  |  |
| --- | --- | --- | --- |
| Opcode | Instruction Name | Register Address (bits) | Immediate (bits) |
| 001101 | ADD | <1> | <9> |
| 001110 | SUB | <1> | <9> |
| 001111 | LSR | <1> | <9> |
| 010000 | LSL | <1> | <9> |
| 010001 | MUL | <1> | <9> |
| 010010 | DIV | <1> | <9> |
| 010011 | MOD | <1> | <9> |
| 010100 | CMP | <1> | <9> |
| 010101 | INC | <1> | 000000000 |
| 010110 | DEC | <1> | 000000000 |
| 010111 | AND | <1> | <9> |
| 011000 | OR | <1> | <9> |
| 011001 | XOR | <1> | <9> |
| 011010 | NOT | <1> | 000000000 |
| 011011 | RSR | <1> | <9> |
| 011100 | RSL | <1> | <9> |
| 011101 | FCT | <1> | <9> |

Move immediate into register instruction:

|  |  |  |  |
| --- | --- | --- | --- |
| Opcode | Instruction Name | Register Address (bits) | Immediate (bits) |
| 011110 | MOV | <1> | <9> |

Exemple:

ADD X => ACC = ACC + X

ADD X, immediate => X = X + immediate

SUB X => ACC = ACC – X

SUB X, immediate => X = X - immediate

LSR X => ACC = ACC >> X

LSR X, immediate => X = X >> immediate

LSL X => ACC = ACC << X

MUL X => ACC = ACC \* X

DIV X => ACC = ACC / X

MOD X => ACC = ACC % X

INC X => X = X + 1

DEC X => X = X - 1

MOV X, immediate => X = immediate

CMP X => compara ACC cu X

Asemanator pentru AND, OR, XOR, RSR, RSL.

Resetting the accumulator can be done like: TRX, SUB X

Factorial: result can be only saved in registers X or Y

Ex: FCT X 3 -> moves in the X register the value of 3! which is 6.