**Formatele instrucțiunilor**

**Categoria 1**

**LDR, STR** (lucrează cu reg. X, Y și acumulator (Acc) )

LDR Rd Rs/imm (Rd = reg. Destinație, Rs = reg. Sursă, imm = val. imediată)

STR imm/Rs Rd

LDR:

|  |  |  |  |
| --- | --- | --- | --- |
| Opcode <5b> | M <1b> | Rd <2b> | Rs / imm <8b> |

STR:

|  |  |  |  |
| --- | --- | --- | --- |
| Opcode <5> | M <1> | Rs / imm <8> | Rd <2> |

**M = modul instrucțiunii; M = 0 ⬄ lucrăm cu regiștrii, M = 1 ⬄ lucrăm cu valoare imediată**

**Reg. X, Y și Acc vor avea adresele 00, 01, respectiv 10.**

Ex. assembly + binar: LDR X Y ⬄ 00000 0 00 00000001 ; LDR X 9 ⬄ 00000 1 00 00001001

**Categoria 2**

**BRZ, BRN, BRC, BRO, BRA, JMP, RET** – salturi

Format:

|  |  |
| --- | --- |
| Opcode <5> | Address <11> ( from PC ) |

Ex. assembly + binar: dacă avem eticheta tag1 la linia 5 dintr-un program, atunci înseamnă că etichetei i se va atribui ADRESA 5.

BRZ tag1 ⬄ 00010 00000000101

**Categoria 3**

**ADD, SUB, LSR, LSL, RSR, RSL, MUL, DIV, MOD, AND, OR, XOR**

Toate instrucțiunile de acest tip (aritmetico-logice) au un registru destinație pentru rezultat și doi regiștrii sursă. Destinația tot timpul e Acc, iar una din surse este tot Acc. Cealaltă sursă poate fi X, Y sau imm.

Format:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Opcode <5> | M <1> | Rd <2> | Rs1 <2> | Rs2 (X/Y/imm) <6> |

Ex. assembly + binar: ADD X ⬄01001 0 10 10 000000 (se execută Acc = Acc + X)

ADD 10 ⬄ 01001 1 10 10 001010 (se execută Acc = Acc + 10)

**Categoria 4**

**MOV, NOT, INC, DEC, CMP** (restul de instrucțiuni rămase)

**MOV** – lucrează cu Acc, X, Y

|  |  |  |  |
| --- | --- | --- | --- |
| Opcode <5> | M <1> | Rd <2> | Rs / imm <8> |

Ex. assembly + binar: MOV X Y ⬄ 10101 0 00 00000001; MOV Y 1 ⬄ 10101 1 01 00000001

**NOT** – lucrează doar cu Acc

|  |  |
| --- | --- |
| Opcode <5> | Adresa Acc <11> |

Ex. assembly + binar: NOT ACC ⬄ 10110 00000000010

**INC, DEC** – lucrează doar cu X și Y

|  |  |
| --- | --- |
| Opcode <5> | Adresa X sau Y <11> |

Ex. assembly + binar: INC X ⬄ 10111 00000000000

**CMP** – lucrează în 4 moduri => 2 biți în câmpul de mod M

CMP X Y – între regiștrii (M = 00)

CMP 3 X – între immediat (primul operand) și un reg (M = 01)

CMP X 3 – între un reg și o val. imediată (al doilea operand) (M = 10)

CMP 5 5 – între două valori imediate (M = 11)

|  |  |  |  |
| --- | --- | --- | --- |
| Opcode <5> | M <2> | Reg / imm <5> | Reg / imm <4> |

Ex. assembly + binar: CMP X Y ⬄ 11001 00 00000 00001

CMP X 3 ⬄ 11001 10 00000 00011

**Operații speciale**

**POW**

|  |  |  |  |
| --- | --- | --- | --- |
| Opcode <5> | M <1> | Reg / imm <5> | Imm <5> |

Reg/imm reprezintă baza, iar ultimii 5 biți (care pot fi doar immediate) sunt exponentul.

Când M = 0 avem conținutul X/Y/Acc ridicat la puterea <<exponent>> și rezultatul se scrie înapoi în registrul bază.

Când M = 1 avem de a face cu o ridicare la putere între 2 valori immediate, iar rezultatul nu mai este scris înapoi.

**NOP**

Operația de NOP are în principiu doar opcode-ul. Dar, atunci când o folosim în contextul unei operații de ridicare la putere în care baza e reprezentată de un registru, avem nevoie ca ultimul NOP din șirul de NOP-uri necesar terminării calculului să conțină informație despre registrul în care trebuie să se scrie rezultatul.

|  |  |  |  |
| --- | --- | --- | --- |
| Opcode <5> | Zero-uri <8> | Reg Dest<2> | WriteEnable <1> |

Ultimul NOP când ridicăm la putere conținutul din registrul Y: 11101 00000000 01 1

|  |  |
| --- | --- |
| **Operație** | **OpCode <5b>** |
| LDR | 00000 |
| STR | 00001 |
| BRZ | 00010 |
| BRN | 00011 |
| BRC | 00100 |
| BRO | 00101 |
| BRA | 00110 |
| JMP | 00111 |
| RET | 01000 |
| ADD | 01001 |
| SUB | 01010 |
| LSR | 01011 |
| LSL | 01100 |
| RSR | 01101 |
| RSL | 01110 |
| MUL | 01111 |
| DIV | 10000 |
| MOD | 10001 |
| AND | 10010 |
| OR | 10011 |
| XOR | 10100 |
| MOV | 10101 |
| NOT | 10110 |
| INC | 10111 |
| DEC | 11000 |
| CMP | 11001 |
| POW | 11010 |
| ??? | 11011 |
| HLT | 11100 |
| NOP | 11101 |