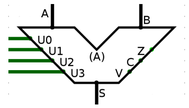
## Instructions

| Nom | Mnémonique | Nb. d'arguments | Opération | Opcode |
|---|---|---|---|---|
| Load A immédiat | LDAi | 1 | A := operande | 0x10 |
| Load A direct | LDAd | 1 | A := RAM[operande] | 0x14 |
| Store A | STA | 1 | RAM[operande] := A | 0x1c |
| Load B immédiat | LDBi | 1 | B := operande | 0x20 |
| Load B direct | LDBd | 1 | B := RAM[operande] | 0x24 |
| Store B | STB | 1 | RAM[operande] := B | 0x2c |
| Add A | ADDA | 0 | A := A + B | 0x30 |
| Add B | ADDB | 0 | B := A + B | 0x34 |
| Sub A | SUBA | 0 | A := A - B | 0x38 |
| Sub B | SUBB | 0 | B := A - B | 0x3c |
| Mul A | MULA | 0 | A := A × B | 0x40 |
| Mul B | MULB | 0 | B := A × B | 0x44 |
| Division de A par 2 | DIVA | 0 | A := A / 2 | 0x48 |
| And A | ANDA | 0 | A := A & B | 0x50 |
| And B | ANDB | 0 | B := A & B | 0x54 |
| Or A | ORA | 0 | A := A \| B | 0x58 |
| Or B | ORB | 0 | B := A \| B | 0x5c |
| Not A | NOTA | 0 | A :=! A | 0x60 |
| Not B | NOTB | 0 | B :=! B | 0x64 |
| Branchement inconditionnel | JMP | 1 | PC := operande | 0x70 |
| Branchement si A nul | JZA | 1 | PC := operande si A = 0 ; PC + 1 sinon | 0x74 |
| Branchement si B nul | JZB | 1 | PC := operande si B = 0 ; PC + 1 sinon | 0x78 |

### Instructions pour la pile

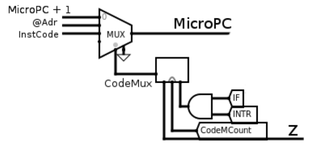
| Nom | Mnémonique | Nb. d'arguments | Opération | Opcode |
|---|---|---|---|---|
| Incrémente le pointeur de pile | INCSP | 0 | SP := SP + 1 | 0x90 |
| Décrémente le pointeur de pile | DECCSP | 0 | SP := SP - 1 | 0x94 |
| Empiler A | PUSHA | 0 | RAM[SP- -] := A | 0xb0 |
| Depiler A | POPA | 0 | A := RAM[++SP] | 0xb4 |
| Sauvegarder A dans la pile | POKEA | 1 | RAM[SP+operande] := A | 0xb8 |
| Récupérer A dans la pile | PEEKA | 1 | A := RAM[SP+operande] | 0xbc |
| Empiler B | PUSHB | 0 | RAM[SP- -] := B | 0xc0 |
| Depiler B | POPB | 0 | B := RAM[++SP] | 0xc4 |
| Sauvegarder B dans la pile | POKEB | 1 | RAM[SP+operande] := B | 0xc8 |
| Récupérer B dans la pile | PEEKB | 1 | B := RAM[SP+operande] | 0xcc |

### Unité Arithématique et Logique (UAL)




| Opcode ($U_3U_2U_1U_0$) | Opération |
|---|---|
| 0000 | S := A |
| 0001 | S := B |
| 0010 | S := A & B |
| 0011 | S := A \| B |
| 0100 | S :=! A |
| 0101 | S :=! B |
| 0110 | S := A + B |
| 0111 | S := A - B |
| 1000 | S := A + 1 |
| 1001 | S := A - 1 |
| 1010 | S := A × B |
| 1011 | S := A >> 1 |

### Multiplexeur du micro-compteur




| CodeMCount | INTR&IF | Z | CodeMux | Opération |
|---|---|---|---|---|
| 000 | x | x | 00 | MicroPC := MicroPC + 1 |
| 001 | x | x | 01 | MicroPC := @Adr |
| 010 | x | x | 10 | MicroPC := InstCode |
| 011 | x | 0 | 00 | MicroPC := MicroPC + 1 |
| 011 | x | 1 | 01 | MicroPC := @Adr |
| 100 | 0 | x | 00 | MicroPC := MicroPC + 1 |
| 100 | 1 | x | 00 | MicroPC := @Adr |

## Sous-programmes

L'appel et le retour de routines (sous-programmes) s'effectuent par les instructions CALL et RET. L'appel de CALL **ne préserve pas** forcément les registres A et B.

L'instruction CALL doit sauvegarder le compteur de programme (PC) sur la pile avant de brancher à l'adresse fournie par l'opérande. L'instruction RET dépile le compteur de programme (PC).

| Nom | Mnémonique | Nb. d'arguments | Opcode |
|---|---|---|---|
| Appel de routine | CALL | 1 | 0xa0 |
| Retour de routine | RET | 0 | 0xa8 |

## Interruptions

L'activation ou inactivation des interruptions dépendents du register *Interrupt Flag* : elles sont activiées si $IF = 1$ et désactivées sinon. L'activation des interruptions s'effectue par l'instruction STI (*Set Interrupt*) et l'inactivation par l'instruction CLI (*Clear Interrupt*).

L'architecture proposée ne supporte qu'une interruption. L'appel de la routine d'interruption s'effectue par l'instruction INT et le retour de la routine d'interruption par l'instruction RTI.

| Nom | Mnémonique | Nb. d'arguments | Opcode |
|---|---|---|---|
| Inactivation des interruptions | CLI | 0 | 0xd0 |
| Activation des interruptions | STI | 0 | 0xd4 |
| Appel de l'interruption | INT | 0 | 0xe0 |
| Retour de l'interruption | RTI | 0 | 0xec |