Crappy CPU machine code equivalence

Antoine Viallon

10 janvier 2020

Table des matières

1 Registers

Available registers :

— `A` : multi-purpose register. Is not overwriten quietly.
— `B` : work register. Used in many operations as a buffer

- U# : user registers. Will **never** be overwritten unless *explicitely* mentionned (see the instructions for more detail). There are only 4 of those currently.
- ret : not directly accessible. Used with CALL and RET
- cmp : not directly accessible. Used with CMP and JMPxxx
- disp : used to display something in decimal. Write-only.
- C : single bit register

2 High level instructions

2.1 ABRT

Set error bit and halt

- ABRT (size : 1, duration : 3)

2.2 ADD

Add a value from a register/memory address/const to a register or a memory address and save it in register A

- ADD A, U# (size : 1, duration : 6)
- ADD R, B (size : 1, duration : 5)
- ADD R, A (size : 1, duration : 6)
- ADD R, @0xHH (size : 2, duration : 9)
- ADD R, #0xHH (size : 2, duration : 7)
- ADD @0xHH, @0xHH (size : 3, duration : 13)
- ADD @0xHH, R (size : 2, duration : 9)
- ADD #0xHH, #0xHH (size : 3, duration : 9)
- ADD @0xHH, #0xHH (size : 3, duration : 10)

2.3 AND

AND two registers, save the result to the first operand

- AND R, B (size : 1, duration : 5)

2.4 CALL

Jump to specified address and save current PC in Ret register. Useful for subroutines.

- CALL #0xHH (size : 2, duration : 8)

2.5 CLR

Clear a register/mem address. Clears B if parameter is an address

- CLR A (size : 1, duration : 6)
- CLR U# (size : 1, duration : 6)
- CLR @0xHH (size : 3, duration : 11)

2.6 CMP

Compare two values (substracts them) and store the result in CMP register. Overwrites B.

- `CMP R, @0xHH` (size : 2, duration : 12)
- `CMP A, B` (size : 1, duration : 8)
- `CMP A, U#` (size : 1, duration : 8)
- `CMP R, #0xHH` (size : 2, duration : 10)
- `CMP U#, A` (size : 1, duration : 8)
- `CMP R` (size : 1, duration : 4)
- `CMP @0xHH` (size : 2, duration : 7)

2.7 DISP

Display a value contained in specified register/memory address as an unsigned integer.

- `DISP R` (size : 1, duration : 4)
- `DISP @0xHH` (size : 1, duration : 7)

2.8 HALT

Halt the CPU

- `HALT` (size : 1, duration : 3)

2.9 INC

Increment register or value at memory address

- `INC R` (size : 1, duration : 5)
- `INC @0xHH` (size : 2, duration : 8)

2.10 JMP

Go to specified address

- `JMP @0xHH` (size : 2, duration : 7)
- `JMP #0xHH` (size : 2, duration : 5)

2.11 JMPBIT

Go to specified address if selected bit of comparison register is 1.

- `JMPBIT %b, #0xHH` (size : 2, duration : 6)

2.12 JMPEQ

Go to specified address if comparison register is zero.

- `JMPEQ #0xHH` (size : 2, duration : 6)

2.13 JMPGE

Go to specified address if comparison register is positive (or zero).

- `JMPGE #0xHH` (size : 2, duration : 6)

2.14 JMPGT

Go to specified address if comparison register is strictly positive.
- `JMPGT #0xHH` (size : 2, duration : 6)

2.15 JMPLE

Go to specified address if comparison register is negative or zero.
- `JMPLE #0xHH` (size : 2, duration : 6)

2.16 JMPLT

Go to specified address if comparison register is strictly negative.
- `JMPLT #0xHH` (size : 2, duration : 6)

2.17 JMPNEQ

Go to specified address if comparison register is NOT zero.
- `JMPNEQ #0xHH` (size : 2, duration : 6)

2.18 JMPPTR

Go to address value at memory address.
- `JMPPTR @0xHH` (size : 2, duration : 7)

2.19 LEDTGL

Toggle led. Useful for debugging.
- `LEDTGL` (size : 1, duration : 4)

2.20 MOV

Move a value from a register/memory address/const to a register or a memory address
- `MOV A, R` (size : 1, duration : 4)
- `MOV B, R` (size : 1, duration : 4)
- `MOV U#, A` (size : 1, duration : 4)
- `MOV U#, B` (size : 1, duration : 4)
- `MOV R, @0xHH` (size : 2, duration : 7)
- `MOV R, #0xHH` (size : 2, duration : 5)
- `MOV @0xHH, R` (size : 2, duration : 6)
- `MOV @0xHH, @0xHH` (size : 3, duration : 10)
- `MOV @0xHH, #0xHH` (size : 3, duration : 7)

2.21 NEG

Compute two's complement of register/memory (useful for substractions), and store result in itself.
- `NEG R` (size : 1, duration : 6)
- `NEG @0xHH` (size : 2, duration : 9)

2.22 NOP

Go to next address
— `NOP` (size : 1, duration : 3)

2.23 NOT

Invert bit by bit register, and store result in itself
— `NOT R` (size : 1, duration : 5)

2.24 OR

OR two registers, save the result to the first operand
— `OR A, B` (size : 1, duration : 5)
— `OR U#, B` (size : 1, duration : 5)

2.25 RET

Revert PC to value saved in Ret register. Use with `CALL`.
— `RET` (size : 1, duration : 6)

2.26 SHIFTL

Shift register to the left
— `SHIFTL R` (size : 1, duration : 5)

2.27 SHIFTR

Shift register to the right
— `SHIFTR R` (size : 1, duration : 5)

2.28 SLEEP

Pause clock for specified amount of ticks
— `SLEEP R` (size : 1, duration : 4)
— `SLEEP #0xHH` (size : 1, duration : 5)
— `SLEEP @0xHH` (size : 1, duration : 7)

2.29 SUB

Sub a value from a memory/register to register A and save it in register A. Overwrites B.
— `SUB R, @0xHH` (size : 2, duration : 12)
— `SUB A, R` (size : 1, duration : 9)

2.30 XOR

XOR two registers, save the result to the first operand
— `XOR A, B` (size : 1, duration : 5)
— `XOR A, A` (size : 1, duration : 6)
— `XOR U#, U#` (size : 1, duration : 6)

3 Instructions (low level)

3.1 default

3.1.1 default : `0x00`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadInstruction`, `incPC`

3.2 ADD

3.2.1 ADD_A_to_A : `0x01`

Micro-instructions :

1. `outA`, `loadB`
2. `enableAdd`, `loadALU`, `outA`
3. `outALU`, `loadA`
4. `clearMIcounter`

3.2.2 ADD_A_to_U0 : `0x02`

Micro-instructions :

1. `outA`, `loadB`
2. `enableAdd`, `loadALU`, `outU0`
3. `outALU`, `loadU0`
4. `clearMIcounter`

3.2.3 ADD_A_to_U1 : `0x03`

Micro-instructions :

1. `outA`, `loadB`
2. `enableAdd`, `loadALU`, `outU1`
3. `outALU`, `loadU1`
4. `clearMIcounter`

3.2.4 ADD_A_to_U2 : `0x04`

Micro-instructions :

1. `outA`, `loadB`
2. `enableAdd`, `loadALU`, `outU2`
3. `outALU`, `loadU2`
4. `clearMIcounter`

3.2.5 ADD_A_to_U3 : `0x05`

Micro-instructions :

1. `outA`, `loadB`
2. `enableAdd`, `loadALU`, `outU3`
3. `outALU`, `loadU3`
4. `clearMIcounter`

3.2.6 ADD_A_to_mem : 0x06

Micro-instructions :

1. outPC, loadRAM
2. outRAM, loadMemAddr, incPC
3. loadRAM, outMemAddr
4. outRAM, loadB
5. enableAdd, loadALU, outA
6. outALU, storeRAM
7. clearMIcounter

3.2.7 ADD_B_to_A : 0x07

Micro-instructions :

1. enableAdd, loadALU, outA
2. outALU, loadA
3. clearMIcounter

3.2.8 ADD_B_to_U0 : 0x08

Micro-instructions :

1. enableAdd, loadALU, outU0
2. outALU, loadU0
3. clearMIcounter

3.2.9 ADD_B_to_U1 : 0x09

Micro-instructions :

1. enableAdd, loadALU, outU1
2. outALU, loadU1
3. clearMIcounter

3.2.10 ADD_B_to_U2 : 0x0a

Micro-instructions :

1. enableAdd, loadALU, outU2
2. outALU, loadU2
3. clearMIcounter

3.2.11 ADD_B_to_U3 : 0x0b

Micro-instructions :

1. enableAdd, loadALU, outU3
2. outALU, loadU3
3. clearMIcounter

3.2.12 ADD_U0_to_A : `0x0c`

Micro-instructions :

1. `outU0`, `loadB`
2. `enableAdd`, `loadALU`, `outA`
3. `outALU`, `loadA`
4. `clearMIcounter`

3.2.13 ADD_U0_to_mem : `0x0d`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadMemAddr`, `incPC`
3. `loadRAM`, `outMemAddr`
4. `outRAM`, `loadB`
5. `enableAdd`, `loadALU`, `outU0`
6. `outALU`, `storeRAM`
7. `clearMIcounter`

3.2.14 ADD_U1_to_A : `0x0e`

Micro-instructions :

1. `outU1`, `loadB`
2. `enableAdd`, `loadALU`, `outA`
3. `outALU`, `loadA`
4. `clearMIcounter`

3.2.15 ADD_U1_to_mem : `0x0f`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadMemAddr`, `incPC`
3. `loadRAM`, `outMemAddr`
4. `outRAM`, `loadB`
5. `enableAdd`, `loadALU`, `outU1`
6. `outALU`, `storeRAM`
7. `clearMIcounter`

3.2.16 ADD_U2_to_A : `0x10`

Micro-instructions :

1. `outU2`, `loadB`
2. `enableAdd`, `loadALU`, `outA`
3. `outALU`, `loadA`
4. `clearMIcounter`

3.2.17 ADD_U2_to_mem : `0x11`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadMemAddr`, `incPC`
3. `loadRAM`, `outMemAddr`
4. `outRAM`, `loadB`
5. `enableAdd`, `loadALU`, `outU2`
6. `outALU`, `storeRAM`
7. `clearMIcounter`

3.2.18 ADD_U3_to_A : `0x12`

Micro-instructions :

1. `outU3`, `loadB`
2. `enableAdd`, `loadALU`, `outA`
3. `outALU`, `loadA`
4. `clearMIcounter`

3.2.19 ADD_U3_to_mem : `0x13`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadMemAddr`, `incPC`
3. `loadRAM`, `outMemAddr`
4. `outRAM`, `loadB`
5. `enableAdd`, `loadALU`, `outU3`
6. `outALU`, `storeRAM`
7. `clearMIcounter`

3.2.20 ADD_const_to_A : `0x14`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadB`, `incPC`
3. `enableAdd`, `loadALU`, `outA`
4. `outALU`, `loadA`
5. `clearMIcounter`

3.2.21 ADD_const_to_U0 : `0x15`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadB`, `incPC`
3. `enableAdd`, `loadALU`, `outU0`
4. `outALU`, `loadU0`
5. `clearMIcounter`

3.2.22 ADD_const_to_U1 : `0x16`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadB`, `incPC`
3. `enableAdd`, `loadALU`, `outU1`
4. `outALU`, `loadU1`
5. `clearMIcounter`

3.2.23 ADD_const_to_U2 : `0x17`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadB`, `incPC`
3. `enableAdd`, `loadALU`, `outU2`
4. `outALU`, `loadU2`
5. `clearMIcounter`

3.2.24 ADD_const_to_U3 : `0x18`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadB`, `incPC`
3. `enableAdd`, `loadALU`, `outU3`
4. `outALU`, `loadU3`
5. `clearMIcounter`

3.2.25 ADD_const_to_const_in_A : `0x19`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadA`, `incPC`
3. `outPC`, `loadRAM`
4. `outRAM`, `loadB`, `incPC`
5. `enableAdd`, `loadALU`, `outA`
6. `outALU`, `loadA`
7. `clearMIcounter`

3.2.26 ADD_const_to_mem : `0x1a`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadMemAddr`, `incPC`
3. `loadRAM`, `outMemAddr`
4. `outRAM`, `loadB`
5. `outPC`, `loadRAM`
6. `enableAdd`, `loadALU`, `outRAM`, `incPC`
7. `outALU`, `storeRAM`
8. `clearMIcounter`

3.2.27 ADD_mem_to_A : `0x1b`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadMemAddr`, `incPC`
3. `loadRAM`, `outMemAddr`
4. `outRAM`, `loadB`
5. `enableAdd`, `loadALU`, `outA`
6. `outALU`, `loadA`
7. `clearMIcounter`

3.2.28 ADD_mem_to_U0 : `0x1c`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadMemAddr`, `incPC`
3. `loadRAM`, `outMemAddr`
4. `outRAM`, `loadB`
5. `enableAdd`, `loadALU`, `outU0`
6. `outALU`, `loadU0`
7. `clearMIcounter`

3.2.29 ADD_mem_to_U1 : `0x1d`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadMemAddr`, `incPC`
3. `loadRAM`, `outMemAddr`
4. `outRAM`, `loadB`
5. `enableAdd`, `loadALU`, `outU1`
6. `outALU`, `loadU1`
7. `clearMIcounter`

3.2.30 ADD_mem_to_U2 : `0x1e`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadMemAddr`, `incPC`
3. `loadRAM`, `outMemAddr`
4. `outRAM`, `loadB`
5. `enableAdd`, `loadALU`, `outU2`
6. `outALU`, `loadU2`
7. `clearMIcounter`

3.2.31 ADD_mem_to_U3 : `0x1f`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadMemAddr`, `incPC`
3. `loadRAM`, `outMemAddr`
4. `outRAM`, `loadB`
5. `enableAdd`, `loadALU`, `outU3`
6. `outALU`, `loadU3`
7. `clearMIcounter`

3.2.32 ADD_mem_to_mem : `0x20`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadMemAddr`
3. `outMemAddr`, `loadRAM`
4. `outRAM`, `loadA`, `incPC`
5. `outPC`, `loadRAM`
6. `outRAM`, `loadMemAddr`
7. `outMemAddr`, `loadRAM`
8. `outRAM`, `loadB`, `incPC`
9. `loadALU`, `enableAdd`, `outA`
10. `outALU`, `storeRAM`
11. `clearMIcounter`

3.3 AND

3.3.1 AND_A_B_to_itself : `0x21`

Micro-instructions :

1. `enableAND`, `loadALU`, `outA`
2. `outALU`, `loadA`
3. `clearMIcounter`

3.3.2 AND_U0_B_to_itself : `0x22`

Micro-instructions :

1. `enableAND`, `loadALU`, `outU0`
2. `outALU`, `loadU0`
3. `clearMIcounter`

3.3.3 AND_U1_B_to_itself : `0x23`

Micro-instructions :

1. `enableAND`, `loadALU`, `outU1`
2. `outALU`, `loadU1`
3. `clearMIcounter`

3.3.4 AND_U2_B_to_itself : `0x24`

Micro-instructions :

1. `enableAND`, `loadALU`, `outU2`
2. `outALU`, `loadU2`
3. `clearMIcounter`

3.3.5 AND_U3_B_to_itself : `0x25`

Micro-instructions :

1. `enableAND`, `loadALU`, `outU3`
2. `outALU`, `loadU3`
3. `clearMIcounter`

3.4 CALL

3.4.1 CALL_addr : `0x26`

Micro-instructions :

1. `outRetAddr`, `loadMemAddr`
2. `outPCp1`, `storeRAM`, `incRet`
3. `outPC`, `loadRAM`
4. `outRAM`, `loadMemAddr`
5. `loadPC`, `cond_always`, `outMemAddr`
6. `clearMIcounter`

3.5 CMP

3.5.1 CMP_A_B : `0x27`

Micro-instructions :

1. `outB`, `enableNOT`, `loadALU`
2. `outALU`, `enableInc`, `loadALU`
3. `outALU`, `loadB`
4. `enableAdd`, `loadALU`, `outA`
5. `outALU`, `loadCmp`
6. `clearMIcounter`

3.5.2 CMP_A_U0 : `0x28`

Micro-instructions :

1. `outU0`, `enableNOT`, `loadALU`
2. `outALU`, `enableInc`, `loadALU`
3. `outALU`, `loadB`
4. `enableAdd`, `loadALU`, `outA`
5. `outALU`, `loadCmp`
6. `clearMIcounter`

3.5.3 CMP_A_U1 : `0x29`

Micro-instructions :

1. `outU1`, `enableNOT`, `loadALU`
2. `outALU`, `enableInc`, `loadALU`
3. `outALU`, `loadB`
4. `enableAdd`, `loadALU`, `outA`
5. `outALU`, `loadCmp`
6. `clearMIcounter`

3.5.4 CMP_A_U2 : `0x2a`

Micro-instructions :

1. `outU2`, `enableNOT`, `loadALU`
2. `outALU`, `enableInc`, `loadALU`
3. `outALU`, `loadB`
4. `enableAdd`, `loadALU`, `outA`
5. `outALU`, `loadCmp`
6. `clearMIcounter`

3.5.5 CMP_A_U3 : `0x2b`

Micro-instructions :

1. `outU3`, `enableNOT`, `loadALU`
2. `outALU`, `enableInc`, `loadALU`
3. `outALU`, `loadB`
4. `enableAdd`, `loadALU`, `outA`
5. `outALU`, `loadCmp`
6. `clearMIcounter`

3.5.6 CMP_A_const : `0x2c`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadB`, `incPC`
3. `outB`, `enableNOT`, `loadALU`
4. `outALU`, `enableInc`, `loadALU`
5. `outALU`, `loadB`
6. `enableAdd`, `loadALU`, `outA`
7. `outALU`, `loadCmp`
8. `clearMIcounter`

3.5.7 CMP_A_mem : `0x2d`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadMemAddr`, `incPC`
3. `outMemAddr`, `loadRAM`
4. `outRAM`, `loadB`
5. `outB`, `enableNOT`, `loadALU`
6. `outALU`, `enableInc`, `loadALU`
7. `outALU`, `loadB`
8. `enableAdd`, `loadALU`, `outA`
9. `outALU`, `loadCmp`
10. `clearMIcounter`

3.5.8 CMP_U0_A : `0x2e`

Micro-instructions :

1. `outA`, `enableNOT`, `loadALU`
2. `outALU`, `enableInc`, `loadALU`
3. `outALU`, `loadB`
4. `enableAdd`, `loadALU`, `outU0`
5. `outALU`, `loadCmp`
6. `clearMIcounter`

3.5.9 CMP_U0_const : `0x2f`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadB`, `incPC`
3. `outB`, `enableNOT`, `loadALU`
4. `outALU`, `enableInc`, `loadALU`
5. `outALU`, `loadB`
6. `enableAdd`, `loadALU`, `outU0`
7. `outALU`, `loadCmp`
8. `clearMIcounter`

3.5.10 CMP_U0_mem : `0x30`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadMemAddr`, `incPC`
3. `outMemAddr`, `loadRAM`
4. `outRAM`, `loadB`
5. `outB`, `enableNOT`, `loadALU`
6. `outALU`, `enableInc`, `loadALU`
7. `outALU`, `loadB`
8. `enableAdd`, `loadALU`, `outU0`
9. `outALU`, `loadCmp`
10. `clearMIcounter`

3.5.11 CMP_U1_A : `0x31`

Micro-instructions :

1. `outA`, `enableNOT`, `loadALU`
2. `outALU`, `enableInc`, `loadALU`
3. `outALU`, `loadB`
4. `enableAdd`, `loadALU`, `outU1`
5. `outALU`, `loadCmp`
6. `clearMIcounter`

3.5.12 CMP_U1_const : `0x32`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadB`, `incPC`
3. `outB`, `enableNOT`, `loadALU`
4. `outALU`, `enableInc`, `loadALU`
5. `outALU`, `loadB`
6. `enableAdd`, `loadALU`, `outU1`
7. `outALU`, `loadCmp`
8. `clearMIcounter`

3.5.13 CMP_U1_mem : `0x33`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadMemAddr`, `incPC`
3. `outMemAddr`, `loadRAM`
4. `outRAM`, `loadB`
5. `outB`, `enableNOT`, `loadALU`
6. `outALU`, `enableInc`, `loadALU`
7. `outALU`, `loadB`
8. `enableAdd`, `loadALU`, `outU1`
9. `outALU`, `loadCmp`
10. `clearMIcounter`

3.5.14 CMP_U2_A : `0x34`

Micro-instructions :

1. `outA`, `enableNOT`, `loadALU`
2. `outALU`, `enableInc`, `loadALU`
3. `outALU`, `loadB`
4. `enableAdd`, `loadALU`, `outU2`
5. `outALU`, `loadCmp`
6. `clearMIcounter`

3.5.15 CMP_U2_const : `0x35`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadB`, `incPC`
3. `outB`, `enableNOT`, `loadALU`
4. `outALU`, `enableInc`, `loadALU`
5. `outALU`, `loadB`
6. `enableAdd`, `loadALU`, `outU2`
7. `outALU`, `loadCmp`
8. `clearMIcounter`

3.5.16 CMP_U2_mem : `0x36`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadMemAddr`, `incPC`
3. `outMemAddr`, `loadRAM`
4. `outRAM`, `loadB`
5. `outB`, `enableNOT`, `loadALU`
6. `outALU`, `enableInc`, `loadALU`
7. `outALU`, `loadB`
8. `enableAdd`, `loadALU`, `outU2`
9. `outALU`, `loadCmp`
10. `clearMIcounter`

3.5.17 CMP_U3_A : `0x37`

Micro-instructions :

1. `outA`, `enableNOT`, `loadALU`
2. `outALU`, `enableInc`, `loadALU`
3. `outALU`, `loadB`
4. `enableAdd`, `loadALU`, `outU3`
5. `outALU`, `loadCmp`
6. `clearMIcounter`

3.5.18 CMP_U3_const : `0x38`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadB`, `incPC`
3. `outB`, `enableNOT`, `loadALU`
4. `outALU`, `enableInc`, `loadALU`
5. `outALU`, `loadB`
6. `enableAdd`, `loadALU`, `outU3`
7. `outALU`, `loadCmp`
8. `clearMIcounter`

3.5.19 CMP_U3_mem : `0x39`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadMemAddr`, `incPC`
3. `outMemAddr`, `loadRAM`
4. `outRAM`, `loadB`
5. `outB`, `enableNOT`, `loadALU`
6. `outALU`, `enableInc`, `loadALU`
7. `outALU`, `loadB`
8. `enableAdd`, `loadALU`, `outU3`
9. `outALU`, `loadCmp`
10. `clearMIcounter`

3.6 COPY

3.6.1 COPY : `0x3a`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadMemAddr`, `incPC`
3. `outMemAddr`, `loadRAM`
4. `outRAM`, `loadB`
5. `outPC`, `loadRAM`
6. `outRAM`, `loadMemAddr`, `incPC`
7. `storeRAM`, `outB`
8. `clearMIcounter`

3.6.2 COPY_A_to_A : `0x3b`

Micro-instructions :

1. `outA`, `loadA`
2. `clearMIcounter`

3.6.3 COPY__A__to__B : 0x3c

Micro-instructions :

1. outA, loadB
2. clearMIcounter

3.6.4 COPY__A__to__U0 : 0x3d

Micro-instructions :

1. outA, loadU0
2. clearMIcounter

3.6.5 COPY__A__to__U1 : 0x3e

Micro-instructions :

1. outA, loadU1
2. clearMIcounter

3.6.6 COPY__A__to__U2 : 0x3f

Micro-instructions :

1. outA, loadU2
2. clearMIcounter

3.6.7 COPY__A__to__U3 : 0x40

Micro-instructions :

1. outA, loadU3
2. clearMIcounter

3.6.8 COPY__A__to__cmp : 0x41

Micro-instructions :

1. outA, loadCmp
2. clearMIcounter

3.6.9 COPY__B__to__A : 0x42

Micro-instructions :

1. outB, loadA
2. clearMIcounter

3.6.10 COPY__B__to__B : 0x43

Micro-instructions :

1. outB, loadB
2. clearMIcounter

3.6.11 COPY_B_to_U0 : `0x44`

Micro-instructions :

1. `outB`, `loadU0`
2. `clearMIcounter`

3.6.12 COPY_B_to_U1 : `0x45`

Micro-instructions :

1. `outB`, `loadU1`
2. `clearMIcounter`

3.6.13 COPY_B_to_U2 : `0x46`

Micro-instructions :

1. `outB`, `loadU2`
2. `clearMIcounter`

3.6.14 COPY_B_to_U3 : `0x47`

Micro-instructions :

1. `outB`, `loadU3`
2. `clearMIcounter`

3.6.15 COPY_B_to_cmp : `0x48`

Micro-instructions :

1. `outB`, `loadCmp`
2. `clearMIcounter`

3.6.16 COPY_U0_to_A : `0x49`

Micro-instructions :

1. `outU0`, `loadA`
2. `clearMIcounter`

3.6.17 COPY_U0_to_B : `0x4a`

Micro-instructions :

1. `outU0`, `loadB`
2. `clearMIcounter`

3.6.18 COPY_U0_to_cmp : `0x4b`

Micro-instructions :

1. `outU0`, `loadCmp`
2. `clearMIcounter`

3.6.19 COPY_U1_to_A : `0x4c`

Micro-instructions :

1. `outU1`, `loadA`
2. `clearMIcounter`

3.6.20 COPY_U1_to_B : `0x4d`

Micro-instructions :

1. `outU1`, `loadB`
2. `clearMIcounter`

3.6.21 COPY_U1_to_cmp : `0x4e`

Micro-instructions :

1. `outU1`, `loadCmp`
2. `clearMIcounter`

3.6.22 COPY_U2_to_A : `0x4f`

Micro-instructions :

1. `outU2`, `loadA`
2. `clearMIcounter`

3.6.23 COPY_U2_to_B : `0x50`

Micro-instructions :

1. `outU2`, `loadB`
2. `clearMIcounter`

3.6.24 COPY_U2_to_cmp : `0x51`

Micro-instructions :

1. `outU2`, `loadCmp`
2. `clearMIcounter`

3.6.25 COPY_U3_to_A : `0x52`

Micro-instructions :

1. `outU3`, `loadA`
2. `clearMIcounter`

3.6.26 COPY_U3_to_B : `0x53`

Micro-instructions :

1. `outU3`, `loadB`
2. `clearMIcounter`

3.6.27 COPY_U3_to_cmp : `0x54`

Micro-instructions :

1. `outU3`, `loadCmp`
2. `clearMIcounter`

3.6.28 COPY_mem_to_cmp : `0x55`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadMemAddr`, `incPC`
3. `outMemAddr`, `loadRAM`
4. `outRAM`, `loadCmp`
5. `clearMIcounter`

3.7 DISPLAY

3.7.1 DISPLAY_A : `0x56`

Micro-instructions :

1. `outA`, `loadDisplay`
2. `clearMIcounter`

3.7.2 DISPLAY_B : `0x57`

Micro-instructions :

1. `outB`, `loadDisplay`
2. `clearMIcounter`

3.7.3 DISPLAY_U0 : `0x58`

Micro-instructions :

1. `outU0`, `loadDisplay`
2. `clearMIcounter`

3.7.4 DISPLAY_U1 : `0x59`

Micro-instructions :

1. `outU1`, `loadDisplay`
2. `clearMIcounter`

3.7.5 DISPLAY_U2 : `0x5a`

Micro-instructions :

1. `outU2`, `loadDisplay`
2. `clearMIcounter`

3.7.6 DISPLAY_U3 : `0x5b`

Micro-instructions :

1. `outU3`, `loadDisplay`
2. `clearMIcounter`

3.7.7 DISPLAY_mem : `0x5c`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadMemAddr`, `incPC`
3. `loadRAM`, `outMemAddr`
4. `outRAM`, `loadDisplay`
5. `clearMIcounter`

3.8 FAIL

3.8.1 FAIL : `0x5d`

Micro-instructions :

1. `error`, `halt`

3.9 HALT

3.9.1 HALT : `0x5e`

Micro-instructions :

1. `halt`

3.10 INC

3.10.1 INC_A : `0x5f`

Micro-instructions :

1. `outA`, `enableInc`, `loadALU`
2. `outALU`, `loadA`
3. `clearMIcounter`

3.10.2 INC_B : `0x60`

Micro-instructions :

1. `outB`, `enableInc`, `loadALU`
2. `outALU`, `loadB`
3. `clearMIcounter`

3.10.3 INC_U0 : `0x61`

Micro-instructions :

1. `outU0`, `enableInc`, `loadALU`
2. `outALU`, `loadU0`
3. `clearMIcounter`

3.10.4 INC_U1 : `0x62`

Micro-instructions :

1. `outU1`, `enableInc`, `loadALU`
2. `outALU`, `loadU1`
3. `clearMIcounter`

3.10.5 INC_U2 : `0x63`

Micro-instructions :

1. `outU2`, `enableInc`, `loadALU`
2. `outALU`, `loadU2`
3. `clearMIcounter`

3.10.6 INC_U3 : `0x64`

Micro-instructions :

1. `outU3`, `enableInc`, `loadALU`
2. `outALU`, `loadU3`
3. `clearMIcounter`

3.10.7 INC_mem : `0x65`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadMemAddr`, `incPC`
3. `outMemAddr`, `loadRAM`
4. `outRAM`, `enableInc`, `loadALU`
5. `outALU`, `storeRAM`
6. `clearMIcounter`

3.11 JMP

3.11.1 JMP_const : `0x66`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadPC`, `cond_always`
3. `clearMIcounter`

3.11.2 JMP_if_eq : `0x67`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `incPC`
3. `outRAM`, `loadPC`, `cond_null`
4. `clearMIcounter`

3.11.3 JMP_if_ge : `0x68`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `incPC`
3. `outRAM`, `loadPC`, `cond_pos`, `cond_null`
4. `clearMIcounter`

3.11.4 JMP_if_gt : `0x69`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `incPC`
3. `outRAM`, `loadPC`, `cond_pos`, `cond_not_null`
4. `clearMIcounter`

3.11.5 JMP_if_le : `0x6a`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `incPC`
3. `outRAM`, `loadPC`, `cond_neg`, `cond_null`
4. `clearMIcounter`

3.11.6 JMP_if_lt : `0x6b`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `incPC`
3. `outRAM`, `loadPC`, `cond_neg`, `cond_not_null`
4. `clearMIcounter`

3.11.7 JMP_if_neq : `0x6c`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `incPC`
3. `outRAM`, `loadPC`, `cond_null`, `invert_cond`
4. `clearMIcounter`

3.11.8 JMP_ptr : `0x6d`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadMemAddr`
3. `outMemAddr`, `loadRAM`
4. `outRAM`, `loadPC`, `cond_always`
5. `clearMIcounter`

3.11.9 JMP_sel_bit_0 : `0x6e`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `incPC`
3. `outRAM`, `loadPC`, `cond_selected_bit`
4. `clearMIcounter`

3.11.10 JMP_sel_bit_1 : `0x6f`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `incPC`
3. `outRAM`, `loadPC`, `cond_selected_bit`, `selector0`
4. `clearMIcounter`

3.11.11 JMP_sel_bit_2 : `0x70`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `incPC`
3. `outRAM`, `loadPC`, `cond_selected_bit`, `selector1`
4. `clearMIcounter`

3.11.12 JMP_sel_bit_3 : `0x71`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `incPC`
3. `outRAM`, `loadPC`, `cond_selected_bit`, `selector0`, `selector1`
4. `clearMIcounter`

3.11.13 JMP_sel_bit_4 : `0x72`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `incPC`
3. `outRAM`, `loadPC`, `cond_selected_bit`, `selector2`
4. `clearMIcounter`

3.11.14 JMP_sel_bit_5 : `0x73`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `incPC`
3. `outRAM`, `loadPC`, `cond_selected_bit`, `selector0`, `selector2`
4. `clearMIcounter`

3.11.15 JMP_sel_bit_6 : `0x74`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `incPC`
3. `outRAM`, `loadPC`, `cond_selected_bit`, `selector1`, `selector2`
4. `clearMIcounter`

3.11.16 JMP_sel_bit_7 : `0x75`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `incPC`
3. `outRAM`, `loadPC`, `cond_selected_bit`, `selector0`, `selector1`, `selector2`
4. `clearMIcounter`

3.12 LED

3.12.1 LED_tgl : `0x76`

Micro-instructions :

1. `flipLed`
2. `clearMIcounter`

3.13 LOAD

3.13.1 LOAD_const_to_A : `0x77`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadA`, `incPC`
3. `clearMIcounter`

3.13.2 LOAD_const_to_B : `0x78`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadB`, `incPC`
3. `clearMIcounter`

3.13.3 LOAD_const_to_U0 : `0x79`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadU0`, `incPC`
3. `clearMIcounter`

3.13.4 LOAD_const_to_U1 : `0x7a`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadU1`, `incPC`
3. `clearMIcounter`

3.13.5 LOAD_const_to_U2 : `0x7b`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadU2`, `incPC`
3. `clearMIcounter`

3.13.6 LOAD_const_to_U3 : `0x7c`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadU3`, `incPC`
3. `clearMIcounter`

3.13.7 LOAD_ptr_to_A : `0x7d`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadMemAddr`, `incPC`
3. `outMemAddr`, `loadRAM`
4. `outRAM`, `loadA`
5. `clearMIcounter`

3.13.8 LOAD_ptr_to_B : `0x7e`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadMemAddr`, `incPC`
3. `outMemAddr`, `loadRAM`
4. `outRAM`, `loadB`
5. `clearMIcounter`

3.13.9 LOAD_ptr_to_U0 : `0x7f`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadMemAddr`, `incPC`
3. `outMemAddr`, `loadRAM`
4. `outRAM`, `loadU0`
5. `clearMIcounter`

3.13.10 LOAD_ptr_to_U1 : `0x80`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadMemAddr`, `incPC`
3. `outMemAddr`, `loadRAM`
4. `outRAM`, `loadU1`
5. `clearMIcounter`

3.13.11 LOAD_ptr_to_U2 : `0x81`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadMemAddr`, `incPC`
3. `outMemAddr`, `loadRAM`
4. `outRAM`, `loadU2`
5. `clearMIcounter`

3.13.12 LOAD_ptr_to_U3 : `0x82`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadMemAddr`, `incPC`
3. `outMemAddr`, `loadRAM`
4. `outRAM`, `loadU3`
5. `clearMIcounter`

3.14 NEG

3.14.1 NEG_A : `0x83`

Micro-instructions :

1. `outA`, `enableNOT`, `loadALU`
2. `outALU`, `enableInc`, `loadALU`
3. `outALU`, `loadA`
4. `clearMIcounter`

3.14.2 NEG_B : `0x84`

Micro-instructions :

1. `outB`, `enableNOT`, `loadALU`
2. `outALU`, `enableInc`, `loadALU`
3. `outALU`, `loadB`
4. `clearMIcounter`

3.14.3 NEG_U0 : `0x85`

Micro-instructions :

1. `outU0`, `enableNOT`, `loadALU`
2. `outALU`, `enableInc`, `loadALU`
3. `outALU`, `loadU0`
4. `clearMIcounter`

3.14.4 NEG_U1 : `0x86`

Micro-instructions :

1. `outU1`, `enableNOT`, `loadALU`
2. `outALU`, `enableInc`, `loadALU`
3. `outALU`, `loadU1`
4. `clearMIcounter`

3.14.5 NEG_U2 : `0x87`

Micro-instructions :

1. `outU2`, `enableNOT`, `loadALU`
2. `outALU`, `enableInc`, `loadALU`
3. `outALU`, `loadU2`
4. `clearMIcounter`

3.14.6 NEG_U3 : `0x88`

Micro-instructions :

1. `outU3`, `enableNOT`, `loadALU`
2. `outALU`, `enableInc`, `loadALU`
3. `outALU`, `loadU3`
4. `clearMIcounter`

3.14.7 NEG_mem : `0x89`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadMemAddr`
3. `outMemAddr`, `loadRAM`, `incPC`
4. `outRAM`, `enableNOT`, `loadALU`
5. `outALU`, `enableInc`, `loadALU`
6. `outALU`, `storeRAM`
7. `clearMIcounter`

3.15 NOP

3.15.1 NOP : `0x8a`

Micro-instructions :

1. `clearMIcounter`

3.16 NOT

3.16.1 NOT_A : `0x8b`

Micro-instructions :

1. `enableNOT`, `loadALU`, `outA`
2. `outALU`, `loadA`
3. `clearMIcounter`

3.16.2 NOT_B : `0x8c`

Micro-instructions :

1. `enableNOT`, `loadALU`, `outB`
2. `outALU`, `loadB`
3. `clearMIcounter`

3.16.3 NOT_U0 : `0x8d`

Micro-instructions :

1. `enableNOT`, `loadALU`, `outU0`
2. `outALU`, `loadU0`
3. `clearMIcounter`

3.16.4 NOT_U1 : `0x8e`

Micro-instructions :

1. `enableNOT`, `loadALU`, `outU1`
2. `outALU`, `loadU1`
3. `clearMIcounter`

3.16.5 NOT_U2 : `0x8f`

Micro-instructions :

1. `enableNOT`, `loadALU`, `outU2`
2. `outALU`, `loadU2`
3. `clearMIcounter`

3.16.6 NOT_U3 : `0x90`

Micro-instructions :

1. `enableNOT`, `loadALU`, `outU3`
2. `outALU`, `loadU3`
3. `clearMIcounter`

3.17 OR

3.17.1 OR_A_B_to_itself : `0x91`

Micro-instructions :

1. `enableOR`, `loadALU`, `outA`
2. `outALU`, `loadA`
3. `clearMIcounter`

3.17.2 OR_U0_B_to_itself : `0x92`

Micro-instructions :

1. `enableOR`, `loadALU`, `outU0`
2. `outALU`, `loadU0`
3. `clearMIcounter`

3.17.3 OR_U1_B_to_itself : `0x93`

Micro-instructions :

1. `enableOR`, `loadALU`, `outU1`
2. `outALU`, `loadU1`
3. `clearMIcounter`

3.17.4 OR_U2_B_to_itself : `0x94`

Micro-instructions :

1. `enableOR`, `loadALU`, `outU2`
2. `outALU`, `loadU2`
3. `clearMIcounter`

3.17.5 OR_U3_B_to_itself : `0x95`

Micro-instructions :

1. `enableOR`, `loadALU`, `outU3`
2. `outALU`, `loadU3`
3. `clearMIcounter`

3.18 RET

3.18.1 RET : `0x96`

Micro-instructions :

1. `decRet`
2. `outRetAddr`, `loadRAM`
3. `outRAM`, `loadPC`, `cond_always`
4. `clearMIcounter`

3.19 SHIFTL

3.19.1 SHIFTL_A : `0x97`

Micro-instructions :

1. `outA`, `enableSHIFTL`, `loadALU`
2. `outALU`, `loadA`
3. `clearMIcounter`

3.19.2 SHIFTL_B : `0x98`

Micro-instructions :

1. `outB`, `enableSHIFTL`, `loadALU`
2. `outALU`, `loadB`
3. `clearMIcounter`

3.19.3 SHIFTL_U0 : `0x99`

Micro-instructions :

1. `outU0`, `enableSHIFTL`, `loadALU`
2. `outALU`, `loadU0`
3. `clearMIcounter`

3.19.4 SHIFTL_U1 : `0x9a`

Micro-instructions :

1. `outU1`, `enableSHIFTL`, `loadALU`
2. `outALU`, `loadU1`
3. `clearMIcounter`

3.19.5 SHIFTL_U2 : `0x9b`

Micro-instructions :

1. `outU2`, `enableSHIFTL`, `loadALU`
2. `outALU`, `loadU2`
3. `clearMIcounter`

3.19.6 SHIFTL_U3 : `0x9c`

Micro-instructions :

1. `outU3`, `enableSHIFTL`, `loadALU`
2. `outALU`, `loadU3`
3. `clearMIcounter`

3.20 SHIFTR

3.20.1 SHIFTR_A : `0x9d`

Micro-instructions :

1. `outA`, `enableSHIFTR`, `loadALU`
2. `outALU`, `loadA`
3. `clearMIcounter`

3.20.2 SHIFTR_B : `0x9e`

Micro-instructions :

1. `outB`, `enableSHIFTR`, `loadALU`
2. `outALU`, `loadB`
3. `clearMIcounter`

3.20.3 SHIFTR_U0 : `0x9f`

Micro-instructions :

1. `outU0`, `enableSHIFTR`, `loadALU`
2. `outALU`, `loadU0`
3. `clearMIcounter`

3.20.4 SHIFTR_U1 : `0xa0`

Micro-instructions :

1. `outU1`, `enableSHIFTR`, `loadALU`
2. `outALU`, `loadU1`
3. `clearMIcounter`

3.20.5 SHIFTR_U2 : `0xa1`

Micro-instructions :

1. `outU2`, `enableSHIFTR`, `loadALU`
2. `outALU`, `loadU2`
3. `clearMIcounter`

3.20.6 SHIFTR_U3 : `0xa2`

Micro-instructions :

1. `outU3`, `enableSHIFTR`, `loadALU`
2. `outALU`, `loadU3`
3. `clearMIcounter`

3.21 SLEEP

3.21.1 SLEEP_A : `0xa3`

Micro-instructions :

1. `outA`, `loadSleep`, `incPC`
2. `clearMIcounter`

3.21.2 SLEEP_B : `0xa4`

Micro-instructions :

1. `outB`, `loadSleep`, `incPC`
2. `clearMIcounter`

3.21.3 SLEEP_U0 : `0xa5`

Micro-instructions :

1. `outU0`, `loadSleep`, `incPC`
2. `clearMIcounter`

3.21.4 SLEEP_U1 : `0xa6`

Micro-instructions :

1. `outU1`, `loadSleep`, `incPC`
2. `clearMIcounter`

3.21.5 SLEEP_U2 : `0xa7`

Micro-instructions :

1. `outU2`, `loadSleep`, `incPC`
2. `clearMIcounter`

3.21.6 SLEEP_U3 : `0xa8`

Micro-instructions :

1. `outU3`, `loadSleep`, `incPC`
2. `clearMIcounter`

3.21.7 SLEEP_const : `0xa9`

Micro-instructions :

1. `outPC`, `loadRAM`
2. `outRAM`, `loadSleep`, `incPC`
3. `clearMIcounter`