Micro-operations整理

取指令（所有后续之前）：

t1: MAR ← (PC) //PC = Program Counter

t2: MBR ← Memory

PC ← (PC) +1

t3: IR ← (MBR)

1. STORE X，ACC → [X]

t1: MAR ← (IR(Address))

t2: MBR ← (ACC)

t3: Memory ← (MBR)

2. LOAD X, [X] → ACC

t1: MAR ← (IR(Address))

t2: MBR ← Memory

t3: BR ← (MBR)

ACC ← 0

t4: ACC ← (ACC) + (MBR)

3. ADD X, ACC + [X] → ACC

t1: MAR ← (IR(Address))

t2: MBR ← Memory

t3: BR ← (MBR)

t4: ACC ← (ACC) + (MBR)

4. SUB X, ACC – [X] → ACC

t1: MAR ← (IR(Address))

t2: MBR ← Memory

t3: BR ← (MBR)

t4: ACC ← (ACC) - (MBR)

5. JMPGEZ X, If ACC ≥ 0, then X → PC else PC + 1 → PC

t1: Test ACC and PC ← IR(Address) if ACC = 0

6. JMP X, X → PC

t1: PC ← IR(Address)

7. HALT, Halt a program

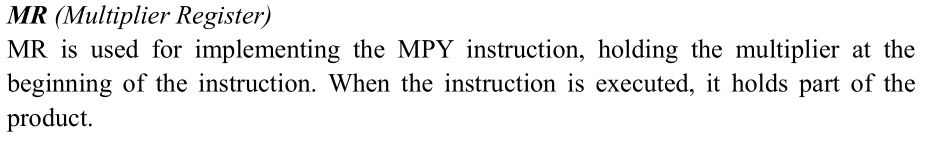
8. MPY X, ACC × [X] → ACC, MR

t1: MAR ← (IR(Address))

t2: MBR ← Memory

t3: BR ← (MBR)

t4: ACC, MR ← (ACC) × (MBR)



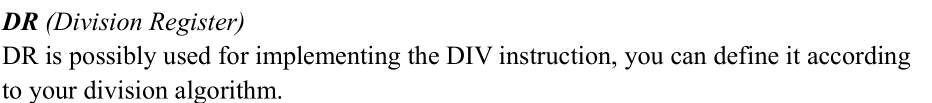
9. DIV X, ACC ÷ [X] → ACC，DR

t1: MAR ← (IR(Address))

t2: MBR ← Memory

t3: BR ← (MBR)

t4: ACC ,DR ← (ACC) ÷ (MBR)



10. AND X，ACC and [X] → ACC

t1: MAR ← (IR(Address))

t2: MBR ← Memory

t3: BR ← (MBR)

t4: ACC ← (ACC) AND (MBR)

11. OR X, ACC or [X] → ACC

t1: MAR ← (IR(Address))

t2: MBR ← Memory

t3: BR ← (MBR)

t4: ACC ← (ACC) OR (MBR)

12. NOT X, NOT X → ACC

t1: MAR ← (IR(Address))

t2: MBR ← Memory

t3: BR ← (MBR)

t4: ACC ← NOT (MBR)

13. SHIFTR, SHIFT ACC to Right 1bit, Logic Shift

t1: ACC ← Shift ACC to Right 1 bit

14. SHIFTL, SHIFT ACC to Left 1bit, Logic Shift

t1: ACC ← Shift ACC to Left 1 bit