Opcode คำสั่ง และ ความหมาย

00000 NOPE ไม่ต้องทำงานอะไร ข้ามไปทำงานคำสั่งถัดไป

00001 accA ⟵ Operand นำค่า 8 bits ของ operand ไปเก็บไว้ใน accA

00010 accB ⟵ Operand นำค่า 8 bits ของ operand ไปเก็บไว้ใน accB

00011 accA ⟵ accB นำค่าจาก accB ไปเก็บไว้ใน accA

00100 accB ⟵ accA นำค่าจาก accA ไปเก็บไว้ใน accB

00101 regC ⟵ accA นำค่าจาก accA ไปเก็บไว้ใน regC

00110 accA ⟵ regC นำค่าจาก regC ไปเก็บไว้ใน accA

00111 regC ⟵ M นำค่าจาก input M ไปเก็บไว้ใน regC

01000 regC ⟵ N นำค่าจาก input N ไปเก็บไว้ใน regC

01001 accA ⟵ pRAM_adr[Operand] นำค่าจาก pRAM ใน address ที่ระบุโดย operand ไปเก็บไว้ใน accA โดยจัดเก็บเฉพาะ 8 bits ด้านขวาเท่านั้น

01010 accA ⟵ rRAM_adr[Operand] นำค่าจาก rRAM ใน address ที่ระบุโดย operand ไปเก็บไว้ใน accA

01011 rRAM_adr[Operand] ⟵ accA นำค่าจาก accA ไปเก็บไว้ใน rRAM ใน address ที่ระบุโดย operand

01100 Jump to address Operand ข้ามไปทำคำสั่งใน address ตามที่ระบุใน Operand แบบไม่มีเงื่อนไข

01101 Jump to address Operand if eq ข้ามไปทำคำสั่งใน address ตามที่ระบุใน Operand ถ้า equal flag เป็น 1

01110 Jump to address Operand if gr ข้ามไปทำคำสั่งใน address ตามที่ระบุใน Operand ถ้า greater flag เป็น 1

01111 Jump to address Operand if le ข้ามไปทำคำสั่งใน address ตามที่ระบุใน Operand ถ้า lesser flag เป็น 1

10000 Jump to address Operand if eq or gr ข้ามไปทำคำสั่งใน address ตามที่ระบุใน Operand ถ้า equal flag หรือ greater flag เป็น 1

10001 accA ⟵ accA + accB นำค่าจาก accA มาบวกกับ accB แล้วไปเก็บที่ accA (ไม่ต้องสนใจกรณีผลบวกเกินขอบเขต) คำนวณแบบ 2's complement 10010 accA ⟵ accA - accB นำค่าจาก accA มาลบกับ accB แล้วไปเก็บที่ accA (ไม่ต้องสนใจกรณีผลลบเกินขอบเขต) คำนวณแบบ 2's complement

10011 accA ⟵ accA * accB นำค่าจาก accA[3..0] มาคูณกับ accB[3..0] แล้วไปเก็บที่ accA (คิด 4 bits ดังนั้นจะมองเป็นเลขบวกอย่างเดียว)

10100 accA ⟵ accA / accB นำค่าจาก accA คิดแบบ binary ไม่ดู signed bit มาหารกับ accB แล้วไปเก็บที่ accA

10101 accA ⟵ accA % accB นำค่าจาก accA คิดแบบ binary ไม่ดู signed bit มา mod กับ accB แล้วไปเก็บที่ accA

10110 accA ⟵ accA ^ accB นำค่าจาก accA[2..0] มายกกำลัง accB[2..0] แล้วไปเก็บที่ accA (ไม่ต้องสนใจกรณีผลลัพธ์เกินขอบเขต)

10111 accA CMP accB เทียบค่า accA กับ accB คำนวณแบบ 2's complement - ถ้า accA == accB ค่า equal flag จะเป็น 1 - ถ้า accA > accB ค่า greater flag จะเป็น 1 - ถ้า accA < accB ค่า lesser flag จะเป็น 1

11000 accA ⟵ NOT(accA) กลับบิตของ accA แล้วเก็บไว้ที่ accA

11001 accA ⟵ accA AND accB นำค่าจาก accA มา bitwise AND กับ accB แล้วไปเก็บที่ accA

11010 accA ⟵ accA OR accB นำค่าจาก accA มา bitwise OR กับ accB แล้วไปเก็บที่ accA 11011 accA ⟵ accA XOR accB นำค่าจาก accA มา bitwise XOR กับ accB แล้วไปเก็บที่ accA

11100 accA ⟵ accA << accB นำค่าจาก accA มา logical shift left ตามค่าของ accB[2..0] แล้วไปเก็บที่ accA โดย shift left ไม่เกิน 7 bits

11101 isPrime(accA) ตรวจสอบว่า accA เป็นจำนวนเฉพาะหรือไม่ ถ้า - accA เป็นจำนวนเฉพาะ ค่า equal flag จะเป็น 1 ให้คิด accA แบบเลขฐาน 2 ปกติ ไม่มีเลขลบ

11110 rRAM[0x0E:0x0F] ⟵ LCM(rRAM_adr[Operand [7:4]] , rRAM_adr[Operand [3:0]]) คำนวณหาค่าคูณร่วมน้อย LCM(m,n) โดย m คือค่าใน rRAM ตำแหน่งตาม operand[7:4] n คือค่าใน rRAM ตำแหน่งตาม operand[3:0] ให้คิดตัวเลขแบบ binary เป็นจำนวนเต็มบวกเท่านั้น นำผลลัพธ์ที่ได้เก็บไว้ที่ rRAM[0x0E] และ rRAM[0x0F] โดยค่า tmost significant บิต result[15:8] เก็บที่ rRAM[0x0E] โดยค่า least significant บิต result[7:0] เก็บที่ rRAM[0x0F] https://en.wikipedia.org/wiki/Least_common_multiple

11111 STOP หยุดการทำงาน ไม่ต้องทำคำสั่งถัดไป แล้วรอสัญญาณ result