

## CEDT Final Project 2567 Digital Logic 2110252: Mini CPU

จะเปิดให้ส่งตัว final test พร้อมกับ future operation

วันพุธที่ 23 ตุลาคม 2567 เวลา 14:00

5 กลุ่มแรกที่ทำแบบ multiple cycle / pipeline (ทำแบบ single clock CPU จะไม่นับใน bonus นี้) และผ่านทุกเคส จะได้ S ทั้งกลุ่ม (โดยไม่ต้องเข้าสอบก็ได้)

คนที่พัฒนาหลัก 2 คนต่อกลุ่มจะได้ S\*

**ขอแสดงความยินดีกับทั้ง 5 กลุ่มด้วยนะครับ กลุ่มที่ทำเสร็จถัดมาก็คือเก่งมาก  
ช้ากว่าแต่ไม่กีนานที่เอง 😊**

**ทั้ง 5 กลุ่มที่ได้ Extra credit**

- ได้ S ทันทีทุกคน
- คนที่เป็นผู้พัฒนาหลักได้ S\*
- ไม่ต้องเข้าสอบพรุ่งนี้ก็ได้ (แต่ถ้าใครอยากเข้าสอบเพื่อได้คะแนนสอบยอดเยี่ยมก็ได้)
- ไม่ต้องฟรีเซนต์ ไม่ต้องทำรายงาน (แต่ถ้าทำมาแล้วก็ส่งได้นะ) → ถ้าจะไม่มาฟรีเซนต์ ให้ลบที่จองออกด้วยนะครับ → ถือว่าส่วนโครงการได้เต็ม 35
- ปกติทางภาควิชาจะมีการมอบประกาศนียบัตรกับนิสิตที่ได้คะแนน Top 3 ของแต่ละวิชาด้วย ดังนั้นถ้านิสิตอยากจะทำคะแนนให้ได้สูงๆ เพื่อลุ้นประกาศนียบัตรนี้ ควรเข้าสอบด้วย

เขียนชื่อกลุ่มและสมาชิก พร้อมระบุคนทำหลัก 2 คน ลงไปในไฟล์ .dig ที่ส่งด้วยนะครับ

ทำเหมือนใส่กรอบเดิม label นั้นแหละ

ข้อมูล testcase แบบ full

[https://docs.google.com/spreadsheets/d/1TIUC1whxUTNDk\\_5jwqrP4Xb7SYOda\\_58/edit?usp=sharing&oid=115188683733264074142&rtpof=true&sd=true](https://docs.google.com/spreadsheets/d/1TIUC1whxUTNDk_5jwqrP4Xb7SYOda_58/edit?usp=sharing&oid=115188683733264074142&rtpof=true&sd=true)

**ให้จองลำดับการนำเสนอ (เข้าด้วยอีเมลนิสิต)**

<https://docs.google.com/spreadsheets/d/1U4k3qLSoolD2SVIlzdwHmqSKDFMKScWfXDO7qqRIBBg>

กลุ่มไหนทำ verilog มาเองและอยากรนำเสนอเพื่อได้ Extra credit ลงเวลาได้เลยนะครับ

<https://docs.google.com/spreadsheets/d/1U4k3qLSoolD2SVIlzdwHmqSKDFMKScWfXDO7qqRIBBg/edit?gid=186526749#gid=186526749>

ให้นิสิตสร้าง CPU โดยใช้โปรแกรม Digital โดยช่วงแรกรอรับคำสั่งเพื่อโหลดโปรแกรมเข้ามาเก็บใน RAM memory ขนาด 256 x 13 bits (pRAM - Program RAM) แล้วทำตามคำสั่งโดยคำสั่งแรกจะอยู่ที่ address 0x00 ทำไปจนเจอคำสั่งให้หยุด จากนั้นรอนกว่าจะได้รับสัญญาณให้ส่งคำตอบ จึงแสดงผลลัพธ์ที่เก็บไว้ใน RAM memory ขนาด 256 x 8 bits (rRAM - Result RAM) ในตำแหน่ง address 0x00 - 0x0F ออกทาง output และ 7-segment จำนวน 2 ตัว

วงจรมี input คือ

- M ขนาด 8 bits พารามิเตอร์ที่ 1
- N ขนาด 8 bits พารามิเตอร์ที่ 2
- progIN ขนาด 13 bits เป็น data input ที่จะส่งโปรแกรมเข้ามาที่ละคำสั่งจนครบ ก่อนจะให้รอสัญญาณเพื่อเริ่มทำงาน
- reset ขนาด 1 bit เมื่อได้ 1 ให้ reset การทำงานของวงจร และ/หรือ เคลียร์ค่าของ rRAM ให้เป็น 0 ทั้งหมด โดยจะมี clock ให้ทำงานส่วนนี้อย่างน้อย 20 clocks
- progLoad ขนาด 1 bit เมื่อได้ 1 ให้เริ่มอ่านค่าจาก progIN ที่ละคำสั่ง ไปเก็บใน pRAM เริ่มตั้งแต่ address 0x00 ไปจนกระทั่ง สัญญาณ progLoad เป็น 0 ซึ่งคำสั่งที่มากที่สุดจะเป็น 256 คำสั่ง
- start ขนาด 1 bit เมื่อได้ 1 ให้เริ่มทำงานตาม pRAM
- result ขนาด 1 bit เมื่อได้ 1 ให้เริ่มส่งคำตอบที่อยู่ใน rRAM
- clk ขนาด 1 bit เป็นสัญญาณ clock สำหรับใช้ในการให้จังหวะ

ส่วนนี้สำหรับเป็นตัวช่วยในการทำงาน grader ไม่ได้ส่งส่วนนี้ให้โดยตรง

- [pRAM ขนาด 256 x 13 bits] สำหรับเก็บโปรแกรม ที่ได้รับเข้ามา สามารถเลือกเป็นแบบใดก็ได้

วงจรมี output คือ

- valid ขนาด 1 bit เป็น 1 เมื่อทำงานตามโปรแกรมเสร็จเรียบร้อยแล้วส่งคำตอบ
- done ขนาด 1 bit เป็น 1 เมื่อการทำงานเสร็จสิ้นและส่งผลคำตอบเรียบร้อยแล้ว หรือตอนเริ่มต้นครั้งแรกที่พร้อมรับคำสั่ง reset / progLoad / start
- output ขนาด 8 bits เป็นค่า output ที่ได้จากการทำงาน โดยแสดงผลลัพธ์ที่อยู่ใน rRAM เป็นเลขฐาน 16 โดยจะแสดงค่าหลังจากได้รับสัญญาณ result

ส่วนนี้สำหรับเป็นตัวช่วยในการทำงาน grader ไม่ได้อ่านจากส่วนนี้

- [7-segment 2 ตัว] แสดงค่าผลลัพธ์ของ output สำหรับให้ตรวจสอบผลลัพธ์
- [rRAM ขนาด 256 x 8 bits] สำหรับเก็บคำตอบ สามารถเลือกเป็นแบบใดก็ได้

ข้อมูลเพิ่มเติมสำหรับการพัฒนา CPU

- สามารถใช้อุปกรณ์ อะไรก็ได้เพิ่มเติมที่มีอยู่ใน Digital แบบมาตรฐาน
- จำนวนบรรทัดของคำสั่งจะอยู่ในช่วง 1 - 255 คำสั่ง
- คำสั่งโปรแกรมที่ส่งมาให้จัดเก็บเข้า pRAM โดยเมื่อทำงานให้อ่านคำสั่งของโปรแกรมตามลำดับ โดยให้ทำงานตั้งแต่บรรทัดแรกไปจนเจอคำสั่งที่ให้หยุดการทำงาน
- CPU นี้จะใช้การคำนวณแบบ 8 bits เป็นหลัก
- ระบบตัวเลขสำหรับคำนวณทางคณิตศาสตร์จะเป็นแบบ 2's complement ยกเว้นจะระบุเป็นอย่างอื่น
- แต่ละคำสั่ง จะออกแบบให้ใช้จำนวน clock เท่ากันหรือไม่เท่ากันก็ได้
- ต้องมีการออกแบบ register เพื่อเก็บค่าสถานะหรือข้อมูลต่างๆเพิ่มเติมเอง ตามความเหมาะสม
- ไม่จำเป็นต้องทำได้ครบทุกคำสั่ง บาง test case อาจจะใช้เพียงไม่กี่คำสั่งก็ได้
- กำหนดให้ใช้ positive edge clock ในการ Synchronous วงจร

แต่ละคำสั่งจะมีโครงสร้างคำสั่งดังนี้

12	11	10	9	8	7	6	5	4	3	2	1	0
Opcode 5 bits					Operand 8 bits							

โครงสร้างของ instruction ขนาด 13 bits แบ่งเป็น opcode 5 bits และ operand 8 bits

Opcode และคำอธิบายเป็นไปดังตารางด้านล่าง

Opcode	คำสั่ง	ความหมาย
00000	NOPE	ไม่ต้องทำงานอะไร ข้ามไปทำงานคำสั่งถัดไป
00001	$accA \leftarrow \text{Operand}$	นำค่า 8 bits ของ operand ไปเก็บไว้ใน accA
00010	$accB \leftarrow \text{Operand}$	นำค่า 8 bits ของ operand ไปเก็บไว้ใน accB
00011	$accA \leftarrow accB$	นำค่าจาก accB ไปเก็บไว้ใน accA
00100	$accB \leftarrow accA$	นำค่าจาก accA ไปเก็บไว้ใน accB
00101	$regC \leftarrow accA$	นำค่าจาก accA ไปเก็บไว้ใน regC
00110	$accA \leftarrow regC$	นำค่าจาก regC ไปเก็บไว้ใน accA
00111	$regC \leftarrow M$	นำค่าจาก input M ไปเก็บไว้ใน regC
01000	$regC \leftarrow N$	นำค่าจาก input N ไปเก็บไว้ใน regC
01001	$accA \leftarrow pRAM\_adr[\text{Operand}]$	นำค่าจาก pRAM ใน address ที่ระบุโดย operand ไปเก็บไว้ใน accA โดยจัดเก็บเฉพาะ 8 bits ด้านขวาเท่านั้น
01010	$accA \leftarrow rRAM\_adr[\text{Operand}]$	นำค่าจาก rRAM ใน address ที่ระบุโดย operand ไปเก็บไว้ใน accA
01011	$rRAM\_adr[\text{Operand}] \leftarrow accA$	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	ข้ามไปทำคำสั่งใน address ตามที่ระบุใน Operand ถ้า lesser

Opcode	คำสั่ง	ความหมาย
	Operand if le	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	$accA \leftarrow accA + accB$	นำค่าจาก accA มาบวกกับ accB แล้วไปเก็บที่ accA (ไม่ต้องสนใจกรณีผลบวกเกินขอบเขต) คำนวณแบบ 2's complement
10010	$accA \leftarrow accA - accB$	นำค่าจาก accA มาลบกับ accB แล้วไปเก็บที่ accA (ไม่ต้องสนใจกรณีผลลบเกินขอบเขต) คำนวณแบบ 2's complement
10011	$accA \leftarrow accA * accB$	นำค่าจาก accA[3..0] มาคูณกับ accB[3..0] แล้วไปเก็บที่ accA (คิด 4 bits ดังนั้นจะมองเป็นเลขบวกอย่างเดียว)
10100	$accA \leftarrow accA / accB$	นำค่าจาก accA คิดแบบ binary ไม่ดู signed bit มาหารกับ accB แล้วไปเก็บที่ accA
10101	$accA \leftarrow accA \% accB$	นำค่าจาก accA คิดแบบ binary ไม่ดู signed bit มา mod กับ accB แล้วไปเก็บที่ accA
10110	$accA \leftarrow accA ^ accB$	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	$accA \leftarrow NOT(accA)$	กลับบิตของ accA แล้วเก็บไว้ที่ accA
11001	$accA \leftarrow accA AND accB$	นำค่าจาก accA มา bitwise AND กับ accB แล้วไปเก็บที่ accA
11010	$accA \leftarrow accA OR accB$	นำค่าจาก accA มา bitwise OR กับ accB แล้วไปเก็บที่ accA
11011	$accA \leftarrow accA XOR accB$	นำค่าจาก accA มา bitwise XOR กับ accB แล้วไปเก็บที่ accA
11100	$accA \leftarrow accA << accB$	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	$rRAM[0x0E:0x0F] \leftarrow LCM($	คำนวณหาค่าคูณร่วมน้อย LCM(m,n) โดย m คือค่าใน rRAM ตำแหน่งตาม operand[7:4]

Opcode	คำสั่ง	ความหมาย
	rRAM_adr[Operand [7:4] ], rRAM_adr[Operand [3:0] ]	ก คือค่าใน rRAM ตำแหน่งตาม operand[3:0] ให้คิดตัวเลขแบบ binary เป็นจำนวนเต็มบวกเท่านั้น  นำผลลัพธ์ที่ได้เก็บไว้ที่ rRAM[0x0E] และ rRAM[0x0F] โดยค่า most significant บิต result[15:8] เก็บที่ rRAM[0x0E] โดยค่า least significant บิต result[7:0] เก็บที่ rRAM[0x0F]  <a href="https://en.wikipedia.org/wiki/Least_common_multiple">https://en.wikipedia.org/wiki/Least_common_multiple</a>
11111	STOP	หยุดการทำงาน ไม่ต้องทำคำสั่งถัดไป แล้วรอสัญญาณ result

accA, accB, regC คือ Accumulator A, Accumulator B, Register C ตามลำดับโดยมีขนาด 8 bits

### การทำงานในทีม

- ให้ทำงานเป็นกลุ่ม กลุ่มละ 3-4 คน โดยนิสิตสามารถจับกลุ่มกันเองได้
  - กลุ่ม 2 คนก็ได้ พออนุโลม แต่ถ้ามีเพื่อนไม่มีกลุ่ม ก็อาจจะขอให้เพื่อนร่วมกลุ่มด้วย
- นิสิตสามารถปรึกษากันระหว่างกลุ่มได้ แต่ห้ามคัดลอก
- สามารถใช้ generative AI ในการช่วยวิเคราะห์และพัฒนางวงจรได้

### การให้คะแนน (draft)

- เอกสารด้านเทคนิคไม่เกิน 10 หน้า (10 คะแนน)
  - แนวคิดการออกแบบ CPU
  - ASM Chart หรือ FSM Chart
  - การออกแบบและพัฒนาส่วน Data Path
  - การออกแบบและพัฒนาส่วน Control Unit
- ส่วนการนำเสนอ (10 คะแนน)
  - สุ่มตัวแทนที่จะมานำเสนอ แสดงว่าทุกคนควรรู้เรื่องและอธิบายได้
  - ให้จัดทำสไลด์ประกอบการนำเสนอ
  - ระยะเวลานำเสนอ 5-10 นาที
- การทำงานตาม test case (80 คะแนน)
  - ในแต่ละ โปรแกรมทดสอบจะมีความแตกต่างกันทั้งเรื่องจำนวนคำสั่งที่ใช้ บรรทัดของคำสั่ง
  - สำหรับโปรแกรมทดสอบเดียวกัน ก็จะมีหลาย test case เพื่อทดสอบส่งสัญญาณ input เช่น M, N, reset, start, result ในเวลาที่ไม่เหมือนกันเพื่อดูการตอบสนองของวงจรที่สร้างขึ้น
- ทุกๆ สมาชิกที่เกินกว่า 4 คน จะถูกหักคนละ 5 คะแนน (-5 คะแนน)

### Extra Credit

- กลุ่ม (จำนวนไม่เกิน 5 คน) ที่สามารถส่งวงจรผ่าน grader ได้ถูกต้องทุก test case จำนวน 5 กลุ่มแรก (ในกลุ่มต้องทำเอง ห้ามทุจริตเอาของผู้อื่นมาดัดแปลงแล้วส่ง) จะได้รางวัลพิเศษ (ต้องแสดงรหัสและชื่อสมาชิกในไฟล์ .dig และแสดงหน้าที่ของแต่ละคนโดยเป็นคนหลักในการออกแบบและพัฒนาไม่เกิน 2 คน) โดยยี้ดรายชื่อตามไฟล์ครั้งแรกที่ส่งผ่านครบทุก test case
  - ถือว่าสอบผ่านได้ S ทั้งกลุ่ม โดยไม่พิจารณาคะแนนแล็ปและคะแนนสอบ
  - คนที่เป็นหลักในการออกแบบและพัฒนาจำนวนไม่เกิน 2 คนต่อกลุ่ม จะได้ S\* คือใบ Certificate ยอดเยี่ยมประจำวิชา
- กลุ่ม (จำนวนไม่เกิน 5 คน) ที่ทำวงจรผ่านโปรแกรม Digital จนผ่านทุก test case แล้ว สามารถจัดทำ CPU ในรูปแบบโปรแกรม verilog ได้ (โดยทำเองไม่ใช่ export จาก Digital) โดยกลุ่มที่ทำต่อ ยอดส่วนนี้จนสำเร็จจำนวน 3 กลุ่มแรก
  - ถือว่าสอบผ่านได้ S ทั้งกลุ่ม โดยไม่พิจารณาคะแนนแล็ปและคะแนนสอบ
  - คนที่เป็นหลักในการออกแบบและพัฒนาจำนวนไม่เกิน 2 คนต่อกลุ่ม จะได้ S\* คือใบ Certificate ยอดเยี่ยมประจำวิชา

## ตัวอย่างการทำงาน

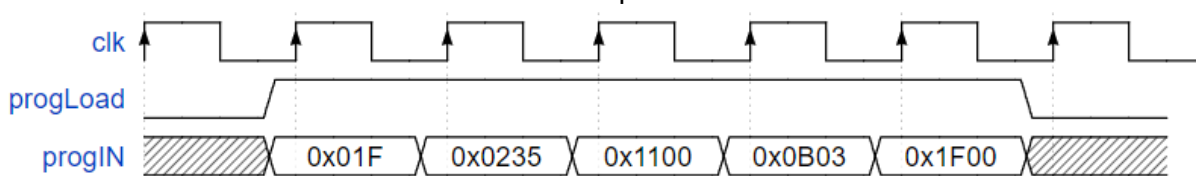
กำหนดให้ข้อมูลโปรแกรมทดสอบเป็นดังนี้

0b 00001 0000 1111	$\text{accA} \leftarrow \text{Operand}$
0b 00010 0011 0101	$\text{accB} \leftarrow \text{Operand}$
0b 10001 0000 0000	$\text{accA} \leftarrow \text{accA} + \text{accB}$
0b 01011 0000 0011	$\text{rRAM\_adr}[\text{Operand}] \leftarrow \text{accA}$
0b 11111 0000 0000	STOP

เริ่มต้นจะมีสัญญาณ reset เพื่อให้วงจรทำการตั้งต้นค่าของระบบ โดยจะมี clock อย่างน้อยจำนวน 20 clock ก่อนจะมีสัญญาณ progLoad

รอสัญญาณ progLoad เป็น 1 จะให้เริ่มรับค่าสิ่งที่ละค่าส่งผ่านทาง progIN ไปเก็บไว้ใน pRAM โดยเริ่มต้นที่ address 0x00 แล้วค่อยๆเพิ่ม address ที่ละ 1 จนกว่าสัญญาณ progLoad เป็น 0 ซึ่งหมายถึง program ได้ถูกส่งให้ครบถ้วนแล้ว

ในกรณีนี้จะส่งมาทั้งหมด 4 คำสั่ง ให้จัดเก็บที่ pRAM address 0x00 - 0x03

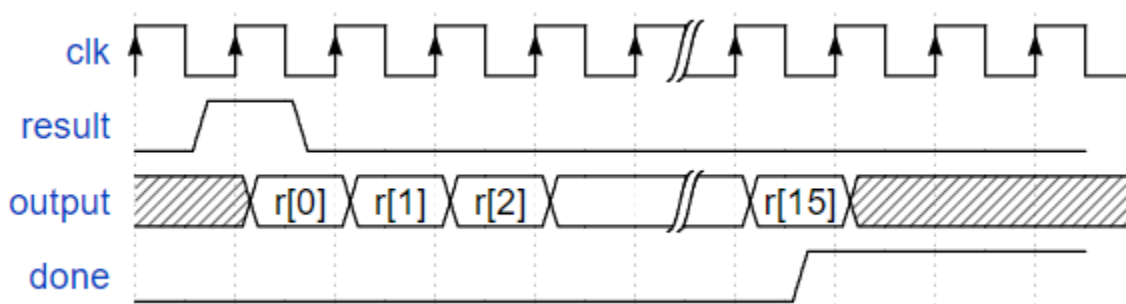


รอสัญญาณ start เป็น 1 จะให้เริ่มทำคำสั่งตามลำดับใน pRAM

เริ่มจากให้

$$\begin{aligned}\text{accA} &\leftarrow 15 \\ \text{accB} &\leftarrow 53 \\ \text{accA} &\leftarrow 15 + 53 = 68 \\ \text{rRAM\_adr}[3] &\leftarrow 68\end{aligned}$$

เมื่อทำงานเสร็จ ให้กำหนดสัญญาณ valid = 1



จากนั้นรอจนได้สัญญาณ result = 1 จึงเริ่มส่งค่าของ rRAM ตั้งแต่ address 0x00 ไปจนถึง 0x0F ซึ่งจะได้

rRAM[0x00]	→ 0
rRAM[0x01]	→ 0
rRAM[0x02]	→ 0
rRAM[0x03]	→ 68 → 0x44

...  
rRAM[0x0F] → 0

เมื่อแสดงผลครบ ให้ส่งสัญญาณ done ออกเป็น 1 เพื่อรอสัญญาณการทำงานใหม่ โดยต้องสามารถรับสัญญาณ reset, progLoad และ start ได้อย่างต่อเนื่อง



## Template

Template สามารถเข้าถึงได้จาก [TEMPLATE V2.dig](https://drive.google.com/file/d/1RSEFD2m7Z0iy-kiXMjbMnvUEdCjVwVkg)  
<https://drive.google.com/file/d/1RSEFD2m7Z0iy-kiXMjbMnvUEdCjVwVkg>

## Testcase

ไฟล์ที่ไว้สร้าง test case สามารถทดลองใช้ได้จาก excel (ให้โหลดลงที่เครื่องมาทดลอง ตัว testcase จะอยู่ที่คอลัมน์ P)

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R										
Program1	Test1																										
M	N	progIN	reset	progLoad	start	result	clk	valid	done	output		Meaning	Assembly	Machine code	M N	progIN	reset	progLoad	start	result	clk	valid	done	output			
0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	testcase format	
0	0	0	0	0	0	0	0	1	0	0	0	clock20 รอให้ reset วจจร			0	0	0	0	0	0	0	0	0	0	0	0 0 0 0 0 0 0 0 0 0 0 0	
0	0	271	0	1	0	0	0	0	0	0	0		accA ← Operand	00001 0000 1111	0	0	271	0	1	0	0	0	0	0	0	0	0 0 0 0 0 0 0 0 1 0 0 0 0 0
0	0	271	0	1	0	0	0	1	0	0	0	โหลดค่าสิ่งที่ 1 เข้า pRAM			0	0	271	0	1	0	0	0	0	0	0	0	0 0 271 0 1 0 0 1 0 0 0 0 0
0	0	565	0	1	0	0	0	0	0	0	0		accB ← Operand	00010 0011 0101	0	0	565	0	1	0	0	0	0	0	0	0	0 0 565 0 1 0 0 0 0 0 0 0 0
0	0	565	0	1	0	0	0	1	0	0	0	โหลดค่าสิ่งที่ 2 เข้า pRAM			0	0	565	0	1	0	0	0	0	0	0	0	0 0 565 0 1 0 0 1 0 0 0 0 0
0	0	4352	0	1	0	0	0	0	0	0	0		accA ← accA + accB	10001 0000 0000	0	0	4352	0	1	0	0	0	0	0	0	0	0 0 4352 0 1 0 0 0 0 0 0 0 0
0	0	4352	0	1	0	0	0	1	0	0	0	โหลดค่าสิ่งที่ 3 เข้า pRAM			0	0	4352	0	1	0	0	0	0	0	0	0	0 0 4352 0 1 0 0 1 0 0 0 0 0
0	0	2819	0	1	0	0	0	0	0	0	0		RAM_add[Operand] ← accA	01011 0000 0011	0	0	2819	0	1	0	0	0	0	0	0	0	0 0 2819 0 1 0 0 0 0 0 0 0 0
0	0	2819	0	1	0	0	0	1	0	0	0	โหลดค่าสิ่งที่ 4 เข้า pRAM			0	0	2819	0	1	0	0	0	0	0	0	0	0 0 2819 0 1 0 0 1 0 0 0 0 0
0	0	7936	0	1	0	0	0	0	0	0	0		STOP	11111 0000 0000	0	0	7936	0	1	0	0	0	0	0	0	0	0 0 7936 0 1 0 0 0 0 0 0 0 0
0	0	7936	0	1	0	0	0	1	0	0	0	โหลดค่าสิ่งที่ 5 เข้า pRAM			0	0	7936	0	1	0	0	0	0	0	0	0	0 0 7936 0 1 0 0 1 0 0 0 0 0
0	0	0	0	0	1	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0 0 0 0 0 1 0 0 0 0 0 0	
0	0	0	0	0	1	0	0	1	0	0	0	รับค่าที่ start ให้เริ่มทำงาน			0	0	0	0	0	0	0	0	0	0	0	0	0 0 0 0 0 1 0 1 0 0 0 0
0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0 0 0 0 0 0 0 0 x 0 0 0
0	0	0	0	0	0	0	0	0	0	0	0		ทำงานตามโปรแกรม		0	0	0	0	0	0	0	0	0	0	0	0	0 0 0 0 0 0 0 1 x 0 0 0
0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0 0 0 0 0 0 0 0 x 0 0 0
0	0	0	0	0	0	0	0	0	0	0	0		ทำงานตามโปรแกรม		0	0	0	0	0	0	0	0	0	0	0	0	0 0 0 0 0 0 0 1 x 0 0 0
0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0 0 0 0 0 0 0 0 x 0 0 0
0	0	0	0	0	0	0	0	0	0	0	0		ทำงานตามโปรแกรม		0	0	0	0	0	0	0	0	0	0	0	0	0 0 0 0 0 0 0 1 x 0 0 0
0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0 0 0 0 0 0 0 0 x 0 0 0

[https://docs.google.com/spreadsheets/d/1TIUC1whxUTNDk\\_5jwqrP4Xb7SYOda\\_58/edit?usp=s](https://docs.google.com/spreadsheets/d/1TIUC1whxUTNDk_5jwqrP4Xb7SYOda_58/edit?usp=s)  
[haring&ouid=115188683733264074142&rtpof=true&sd=true](https://docs.google.com/spreadsheets/d/1TIUC1whxUTNDk_5jwqrP4Xb7SYOda_58/edit?usp=s&haring&ouid=115188683733264074142&rtpof=true&sd=true)

ถ้ามีสิดได้จัดทำ testcase และทดสอบแล้วว่าใช้งานได้สามารถ upload เพื่อมาแชร์กันได้

<https://drive.google.com/drive/folders/1pVPxsRP0wovPDF3y4Q1z42OHkex2D9pj?usp=sharing>

โดยให้ upload ที่เป็น excel ด้วยเพื่อตรวจทานได้สะดวก

โดยตั้งชื่อตามที่เห็นสมควร

พี่เดชได้จัดทำ testcasegenerator.py มาเพิ่มครับ สามารถทดลองใช้กันได้

[https://drive.google.com/file/d/1DnZNXi1npDjflYNbxk54oTro5Eq\\_ip0l/view?usp=sharing](https://drive.google.com/file/d/1DnZNXi1npDjflYNbxk54oTro5Eq_ip0l/view?usp=sharing)

## Test case นะครับ

- Test case ใน grader ที่ให้ทดสอบจะสอดคล้องกับ test case ในไฟล์ program Test.xlsx
- 01\_P01T01 ทดสอบการโหลดโปรแกรมและการส่งค่าทั่วไป
- 02\_P01T02 request result เป็นครั้งที่ 2 หลังจากส่ง output ออกมาแล้ว
- 03\_P02T01 Test simple jump
- 04\_P03T01 pRamLoad\_Test
- 05\_P04T01 isPrime\_Test
- 06\_P05T01 isPrime + CMP
- 07\_P06T01 ทดสอบการอ่านค่า M และ N
- 08\_P06T02 ทดสอบการอ่านค่า M และ N / แล้วก็ทดสอบการ run ใหม่โดยไม่ได้ โหลดโปรแกรมใหม่
- 09\_P07T01 ทดสอบค่าสั่งพิเศษ LCM
- 10\_P07T02 ทดสอบค่าสั่งพิเศษ LCM โดยให้ค่าคำตอบจะเกิน 8 บิต และเก็บตัวต้นไว้ที่ address 0x0F ซึ่งจะไว้เก็บคำตอบด้วย
- 11\_P08T01 ทดสอบการ + - \* / % ^

- 12\_P08T02 ทดสอบการ + - \* / % ^ โดยการสั่งแก้บางส่วนของโปรแกรมแล้วทำงานเลย
- 13\_P09T01 ทดสอบ bitwise operation NOT AND OR XOR <<
- 14\_P09T02 ทดสอบ bitwise operation NOT AND OR XOR << แล้วก็ reset ระหว่างทำงาน

## กำหนดการและแนวทางในการส่ง

### วันประกาศโครงการ

- ให้นิสิตจับกลุ่มและแจ้งรายละเอียดกลุ่ม โดยให้สมาชิกเพียงคนเดียวเป็นตัวแทนในการกรอก (รวมถึงต้องใช้ account นี้ในการ update และส่งข้อมูลตลอด)  
<https://www.mycourseville.com/?q=courseville/worksheet/53675/1425077>
- ให้กรอกชื่อทีมและสมาชิกภายในวันพุธที่ 16 ตุลาคม 2567
- เริ่มออกแบบและพัฒนาวงจร

### ก่อนวันกำหนดส่งระยะหนึ่งภายในวันที่ 18 ตุลาคม 2567

- แจ้งตัวอย่าง test case ที่ใช้ในการทดสอบเบื้องต้น (ซึ่งนิสิตจะทราบคำสั่งที่มี จำนวนคำสั่ง จำนวน clock มากที่สุดที่ยอมให้ใช้)
- เปิด grader ให้ทดลองส่งบาง test case

### ก่อนวันกำหนดส่ง

- ทயอยเพิ่มและแจ้ง Test case (เกือบ) ทั้งหมดที่ใช้ในการทดสอบ
- ให้จองลำดับการนำเสนอ  
<https://docs.google.com/spreadsheets/d/1U4k3qLSoolD2SVIIdzwHmqSKDFMKScWfXDQ7qqRIBBg>

### วันกำหนดส่ง 24 ตุลาคม 2567

#### เวลาประมาณ 12:00 หลังสอบเสร็จ

- เพิ่ม Test case ทั้งหมดที่ใช้ในการทดสอบ
- เริ่มนับเวลาส่งอย่างเป็นทางการเพื่อคัดเลือกทีมที่ส่งได้ครบทุก test case ตาม Extra Credit

### ช่วงการนำเสนอ

- ให้ Upload เอกสารสำหรับนำเสนอให้เสร็จ
- นำเสนอกลุ่มละประมาณ 5-10 นาที
- โดยคนนำเสนอและตอบคำถามจะถูกสุ่มจากสมาชิกคนใดคนหนึ่ง
- นำเสนอทีละกลุ่ม ตามลำดับการจอง

### หลังการนำเสนอ

- มีเวลาให้ทำการพัฒนา CPU เพิ่มเติม
- Grader จะปิดรับการส่งในวันที่ 24 ตุลาคม 2567 ตอนเที่ยงคืน
- ให้ Upload เอกสารต่างๆ เช่นไฟล์สำหรับนำเสนอ รายงาน ไฟล์โปรแกรม และ capture หน้าจอคะแนน grader ผ่านทาง MCV ให้เสร็จก่อนเที่ยงคืน

### หลังวันกำหนดส่ง 26-27 ตุลาคม 2567

- พักผ่อนหรือเตรียมตัวสำหรับวิชาถัดไป
- ขอให้นิสิตทุกคนโชคดีและสนุกกับการเรียน

มีคำถามตรงไหนเพิ่มเติมสามารถ เพิ่ม comment มาได้เลยนะครับ

Comment 1จับ 2 คนได้ไหม → ได้ครับ พออนุโลม แต่อยากให้ เป็น 3-4 คนมากกว่า

Comment 2 Output ขนาด 8 bits จะต้องมีทั้งหมดกี่ตัวครับ จะต้องสร้างให้ครบ 16 ตัวเพื่อแสดงผลค่าใน rRAM ที่ address 0x00 จนถึง 0x0F เลยไหมครับ → ไม่ต้องสร้างตัวแปรให้ครบได้ครับ คือใน rRAM ตอนโดนสัญญาณ clear ก็ให้ล้างค่าเป็น 0 ไว้ก่อน, แล้วพอโดนคำสั่งให้ไปเก็บค่าใน rRAM ก็ค่อยเก็บค่าตามตำแหน่งที่ระบุ → ตอนแสดงผลก็ค่อยๆดึงค่าจาก rRAM เอามาส่งออกให้ทาง output

Comment 3 opcode 01101 - 10000 ค่าของ flag ได้จากการเอาตัวอะไรมาเปรียบเทียบกับหรือครับ → ก็แล้วแต่คำสั่งก่อนหน้าว่าจะเอาอะไรมาเปรียบเทียบ แต่ผลจากการเปรียบเทียบก็จะเอามาใช้สำหรับการ Jump คำสั่ง

- ก็คือขึ้นอยู่กับคำสั่ง 10111 ก่อนหน้าใช่ไหมครับ ถ้าเป็นอย่างนั้นแสดงว่า ในชุดคำสั่งทั้งหมด คำสั่ง 10111 จะมาก่อน 01101-10000 แน่แน่นอนใช่ไหมครับ → ควรมีการเทียบค่าก่อน หรืออาจจะเทียบไว้นานแล้วค่อยมา Jump ก็ได้ครับ หรือบางทีก็คือเรียกคำสั่ง 11101 (isPrime) ก็ได้

Comment 4 acc คืออะไรหรือครับ → accumulator ก็เป็น register ที่เอาไว้ส่งค่าให้ ALU คิดคำนวณครับ แล้วก็รูดค่าจาก ALU เอาไปทำคำสั่งอื่นๆต่อ [https://en.wikipedia.org/wiki/Accumulator\\_\(computing\)](https://en.wikipedia.org/wiki/Accumulator_(computing))

Comment 5 มีโอกาสที่ reset กับ progLoad เป็น 1 พร้อมกันไหมครับ → ไม่มีครับ → test case ที่เหมาะสมจะไม่น่าเป็นเช่นนั้น

- แล้วมีโอกาสที่หลังจาก start แล้ว progLoad จะเป็น 1 ไหมครับ → ก็ต้องขอ result ก่อน แล้วค่อยไปเพื่อ reset / progLoad ได้ครับ

Comment 6 ระหว่างที่ CPU กำลังทำงานอยู่ จะมีโอกาสที่ reset เป็น 0 ไหมครับ → ปกติ reset = 0 คือทำงานไปเรื่อยๆ ไม่ reset ครับ , ถ้าจะให้ reset คือเป็น 1

ซึ่งมีโอกาสที่จะได้รับสัญญาณ reset ระหว่างทำงานได้ แล้วระบบก็ต้องรอพวก progLoad / start ใหม่

- ก็คือหยุดการทำงานทันที (Sync) แบบไม่ต้องรอ Opcode 11111 เลยใช่ไหมครับ → คือเข้าสู่ standby เพื่อรอคำสั่งรอบใหม่ครับ แต่ถ้าเจอ opcode 11111 ก็คือให้รอจังหวะไม่ต้องทำโปรแกรมอะไรเพิ่มเติมแล้ว รอสัญญาณ result เพื่อส่งคำตอบ (แต่ถ้าโดน reset ก็คือไปล้างค่าแล้วรอเริ่มใหม่นะ)

Comment 7 มีโอกาสที่สัญญาณ result จะติดก่อนสัญญาณ valid ไหมครับ → มีครับ ก็คือกรณีที่วงจรที่สร้างมาทำงานช้ากว่าที่กำหนด ระบบจะร้องขอคำตอบแล้วแต่วงจรยังไม่ทัน → เคสนี้ก็คือรันได้ผิด

Comment 8 กรณีที่ Done แล้ว จะต้องรับสัญญาณ Reset, ProgLOAD แล้ว start ตามลำดับเท่านั้นไหมครับ หรือว่า Done แล้ว สามารถ Start ได้เลยไหมครับโดยไม่ต้อง Reset → เป็นไปได้ครับ คือเราอาจจะใช้ program เดิม แต่เปลี่ยน input M กับ N แทน

Comment 9 isPrime(acc A) คิดแบบ Two's complement หรือว่าเลขฐานสองปกติครับ → คิดแบบฐานสองปกติครับ

Comment 10 หลังจาก start เป็น 1 แล้ว ระหว่างรันคำสั่งใน RAM อยู่ มีโอกาสที่ start จะเปลี่ยนค่ามั้ยครับ → start จะเป็น 1 เพียงสั้นๆ อาจจะเพียงแค่ 1 clock

Comment 11 มีกรณีที่ equal flag กับ greater หรือ lesser flag มีค่าเป็น 1 พร้อมกันไหมครับ ถ้าหากใช้คำสั่ง isPrime() แล้วใช้คำสั่ง Compare ต่อ → อันนี้ต้องขึ้นอยู่กับ การ implement ครับ ซึ่งจริงๆไม่ควรให้เป็นอย่างนั้น

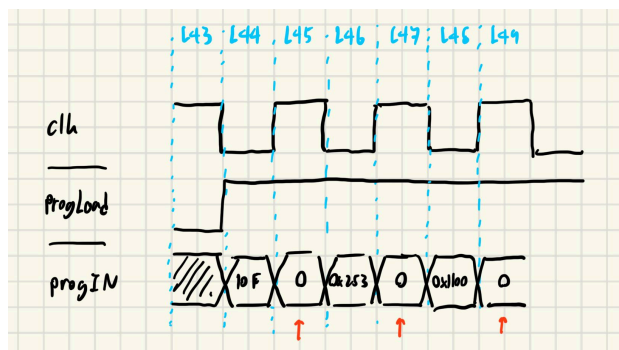
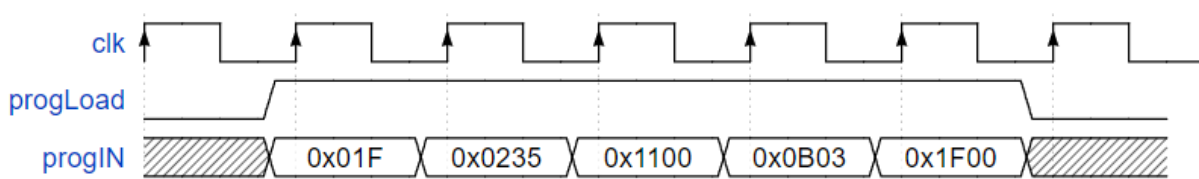
Comment 12 ถ้า greater flag เป็น 1 แล้วจะให้ lesser flag กับ equal flag เป็น 0 ด้วยไหมครับ → ควรเป็นอย่างนั้นครับ

Comment 13 left shift ตามค่า accB คือถ้า accB มีค่าเป็น 4 ให้เลื่อนbitทั้งหมดวนไปทางซ้าย 4 รอบไหมครับ → ใช่ครับ → เป็น shift left ไม่ต้อง rotate shift left

- ค่า accB คิดแบบ binary ปกติหรือว่า Two's complement ครับ → คิดแบบ binary โดยใช้ accB[2..0] จำนวน 3 bits ครับ

Comment 14 reset จะเป็น 1 เพียงสั้นๆ เหมือน comment ที่ 10 ไหมครับ → ใช่ครับ สั้นๆ

Comment 15 ในเอกสาร progIN จะมีค่าทั้งตอน clk 0 และ 1 แต่ใน test case ของ template ค่า progIN มีเฉพาะตอน clk เป็น 0 ควรยึดอันไหนครับ → จริงๆแล้วขอเพียง มีสัญญาณ progIN ที่ถูกต้อง ก่อน clock จะเป็นขาขึ้น ก็พอครับ จังหวะที่ clock=1 ระบบไม่น่าจะได้อ่านอยู่แล้วครับ → พยายามทำระบบให้รองรับการอ่านจังหวะขอบขาขึ้นแล้วกันนะครับ



X 01_P01T01 failed								
	M	N	progIN	reset	progLoad	start	result	clk
L43	0	0	0	0	0	0	0	1
L44	0	0	10F	0	1	0	0	0
L45	0	0	0	0	1	0	0	1
L46	0	0	0x235	0	1	0	0	0
L47	0	0	0	0	1	0	0	1
L48	0	0	0x1100	0	1	0	0	0
L49	0	0	0	0	1	0	0	1

Comment 16 ProgLoad เปลี่ยนเป็น 0 ในช่วงเวลาเดียวกันกับ Start เปลี่ยนเป็น 1 ใหม่ครับ → ใช่ครับ

Comment 17 Isprime(accA) ถ้า accA ไม่ใช่จำนวนเฉพาะ ให้ข้ามไปไม่ต้องทำอะไร (คงค่าทุก Flag ไว้ให้เป็นเหมือนเดิม) หรือให้ตั้งค่า Equal flag เป็น 0 ครับ → ตั้ง equal flag เป็น 0 ครับ

Comment 18 ถ้า greater flag หรือ lesser flag เป็น 1 อยู่ก่อนหน้านี้แล้วเจอคำสั่ง IsPrime(accA) ที่ทำให้ equal flag เป็น 1 ค่าถามคือต้อง reset ทั้งค่า greater flag และ lesser flag เป็น 0 ด้วยไหมครับหรือคงค่าเอาไว้เหมือนเดิม → คงไว้ได้ครับ โปรแกรมที่ควรจะเป็นก็ควรที่เราเรียก isPrime ซึ่งจะกระทบเพียง equal flag อย่างเดียว

Comment 19 ถ้าเดิมที่มี Flag อยู่แล้ว เมื่อเรียกใช้ isPrime(accA) แล้วค่า Flag จะเปลี่ยนเป็นแบบนี้

(GT = 1, EQ = 0, LT = 0) => ไม่เป็นจำนวนเฉพาะ => (GT = 1, EQ = 0, LT = 0)

(GT = 1, EQ = 0, LT = 0) => เป็นจำนวนเฉพาะ => (GT = 1, EQ = 1, LT = 0)

(GT = 0, EQ = 1, LT = 0) => ไม่เป็นจำนวนเฉพาะ => (GT = 0, EQ = 0, LT = 0)

(GT = 0, EQ = 1, LT = 0) => เป็นจำนวนเฉพาะ => (GT = 0, EQ = 1, LT = 0)

Comment 20 Reset progLoad Start result ทั้งสี่ตัวนี้ มีตัวไหนบ้างครับที่อาจจะมีกรณีที่ เป็น 1 พร้อมกันได้ → พร้อมกันไม่ได้เลยครับ testcase ที่ดีจะกำหนดให้เป็น 1 ทีละตัว

Comment 21 เวลาดึงค่าจาก rRam มา output จำเป็นต้องออกมา 16 ตัว ทุกครั้งมั้ย (0x00 - 0x0F) → จังหวะดึงมาแสดงผลตาม result จะแสดง 16 ตัวครับ

Comment 22 ทุกครั้งที่เกิดการ start จำเป็นต้องเสร็จ lesser flag, greater flag และ equal flag เป็น 0 ใหม่ครับ → กระบวนการเคลียร์ตัวแปร ควรจะทำตอนได้รับสัญญาณ reset ครับ

Comment 23 มีโอกาสที่หลังจากขึ้นสัญญาณ done (แสดงค่าครบทั้งหมดแล้ว) แล้วจะให้สัญญาณ result เพื่อแสดงผลค่า output ใหม่อีกไหมครับ → เป็นไปได้ครับ

Comment 24 หากมีการเปลี่ยนค่า start ให้ติดหลังจากโปรแกรมทำงานเสร็จสิ้นแล้ว (หลังมีสัญญาณที่ done) ตาม Comment ที่ 8 จะมีการคงค่าสัญญาณ start ไว้อย่างน้อยกี่ clock ครับ → 1 clock ครับ เหมือนกับการเริ่มปกติ

Comment 25 ทุกครั้งที่กด reset ไม่ว่าจะมีกรณีไหนก็ตามจะมี clock เพื่อให้ 20 clock ทุกครั้งไหมครับ อีกคำถามครับ done จะเปลี่ยนจาก 1 เป็น 0 กรณีไหนบ้างหรือครับ → ใช่ครับเมื่อมี reset จะมีอย่างน้อย 20 clocks

Done จะเปลี่ยนจาก 1 เป็น 0 ตอนที่ได้รับคำสั่ง reset, progLoad และ start (ก็คือการแสดงว่าผลการทำงานเดิมไม่เรียบร้อย)

Comment 26 คำสั่ง 22 ที่ให้ยกกำลังคือเอาแค่ 3 bit แรกของ acc มาคำนวณใช่ไหมคะ → เอา 3 least significant bits ครับ คือ accA[2..0] ไม่ใช่ accA[7..5]

Comment 27 สัญญาณ M และ N จะเป็นสัญญาณสั้น ๆ แบบ start หรือเป็นสัญญาณค้างไว้ตลอดการทำงานครับ ? → ไม่แน่นอนครับ สัญญาณ M และ N นี้จะถูกใช้คู่กับคำสั่ง

00111	regC $\leftarrow$ M	นำค่าจาก input M ไปเก็บไว้ใน regC
01000	regC $\leftarrow$ N	นำค่าจาก input N ไปเก็บไว้ใน regC

แปลว่าถ้าช่วงนั้นเจอคำสั่งเหล่านี้ก็ให้เอาค่าไปเก็บใน regC ช่วงเวลาอื่นถึงค่า M และ N จะเปลี่ยนก็ไม่ต้องสนใจ

Comment 28 สัญญาณ M , N จะถูกใส่ในรอบที่ opcode จะมีการใช้ M , N ใช่หรือไม่ และจะมีโอกาสมีที่คำสั่ง opcode แบบใช้ M,N จะอยู่ติดกันและมี M , N ทั้ง 2 รอบไม่เหมือนกัน จนต้องเก็บค่า M,N ทุกรอบไว้ด้วย  $\rightarrow$  ใช่ครับ เก็บค่าตาม opcode ที่เกี่ยวข้องได้เลย

Comment 29 ฟังก์ชันยกกำลังผลลัพธ์จำเป็นต้องออกมาในรูปแบบของ 2's complement จำนวน 8 bit ใช่ไหมครับ  $\rightarrow$  คิดแบบ binary ได้เลยครับ เพราะว่าเราเอาแค่ 3 bits น้อย มาคำนวณกัน ดังนั้นจะเป็นเลขบวกเสมอ แต่ถ้าเกิดทำมาแล้ว บังเอิญบิตที่ 7 (signed bit) เป็น 1 ก็ปัดทิ้งไป แล้วต่อไปมีการคำนวณทางคณิตศาสตร์ก็คิดแบบ 2's complement ต่อไป (ก็อาจจะผิดพลาดได้ แต่ไม่เป็นไร)

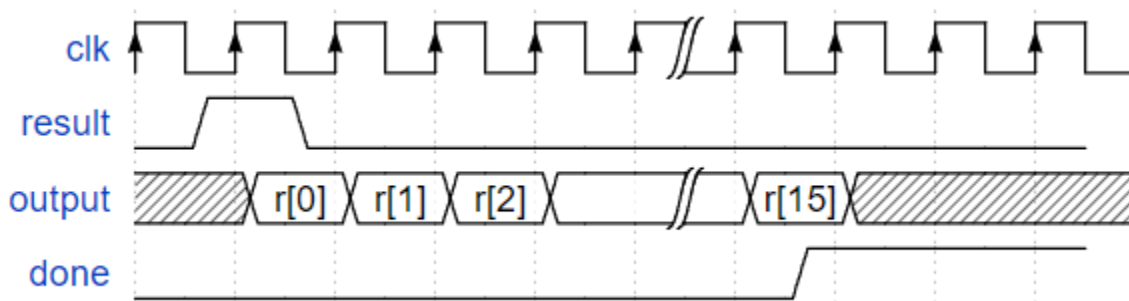
Comment30 ถ้า progLoad เปลี่ยนเป็น 0 แล้ว แต่ start ยังไม่เป็น 1 progLoad มีโอกาสกลับมาเป็น 1 อีกได้มั๊ยคะ  $\rightarrow$  มีโอกาสครับ  $\rightarrow$  แล้วถ้าเป็นอย่างนั้น หลังจากที่ progLoad เป็น 1 รอบที่สอง Instruction set ต่อๆไป จะถูกเก็บไว้ที่ pRAM Address ไหนครับ ระหว่างเริ่มที่ 0x00 ใหม่ แล้ว overwrite อันเก่าไปเลย หรือเก็บที่ Address ถัดจาก Address ล่าสุดที่ได้เคยเก็บไว้ครับ  $\rightarrow$  เก็บใหม่ที่ 0x00 แล้ว overwrite ไปเลย

Comment31 ตอนเขียน Data Path ผมสามารถใช้ tunnel ได้ไหมครับ ?  $\rightarrow$  ใช้ได้ครับ และแนะนำให้ใช้ด้วยครับ เพื่อชีวิตที่ดีครับ

Comment32 reset เป็น synchronous หรือ async ครับ ?  $\rightarrow$  ทำได้ทั้งคู่ครับ

Comment 33 การ Reset ต้องเคลียร์ค่าของ pRAM ด้วยไหมครับ หรือว่าไม่ต้องทำอะไรกับ pRAM  $\rightarrow$  clear แค่ rRam พอครับ

Comment 34 ถ้า done แล้วต้องค้างค่าใน output 0x0F ไว้เหมือนกับ valid แล้วก็ done ใหมครับ  $\rightarrow$  ไม่จำเป็นครับ ขอแค่ช่วงอ่านค่าจาก output ต้องถูกต้องครับ ที่เหลือเป็น dont care ครับ



Comment 35

ที่บอกไม่ต้องสนใจกรณีผลลัพธ์เกินขอบเขตคือยังงั้นหรือครับ แบบให้ remain ค่า accA เหมือนเดิมรีบาวครับ → คือค่าผลลัพธ์ถ้าเกิน 8 bits ก็ไม่ต้องสนใจ ขอให้ 8 bits ถูกต้องก็พอ แล้วก็เก็บไว้ใน accA ต่อ

10110	$accA \leftarrow accA \wedge accB$	นำค่าจาก accA[2..0] มายกกำลัง accB[2..0] แล้วไปเก็บที่ accA (ไม่ต้องสนใจกรณีผลลัพธ์เกินขอบเขต)
-------	------------------------------------	--

Comment 36 ในกรณีที่ Done แล้ว progload เป็น 1 ต่อ ตำแหน่งที่ต้องการจะเก็บค่าจะต้องอยู่ในตำแหน่งไหนหรือครับ → เก็บที่ 0x00 ใน pRAM เลยครับ

Comment 37 ถ้ามีการ reset มาก่อน แล้วหลังจากนั้นก็ให้ progLoad เป็น 1 แล้วจะให้เริ่มเขียนคำสั่งที่ตำแหน่งเริ่มต้นใหม่ หรือตำแหน่งถัดจากคำสั่งล่าสุดครับ → ตำแหน่งตั้งต้นที่ 0x00 ใน pRAM เลยครับ

Comment 38 สัญญาณ M, N จะได้มาตอนที่โหลด progin หรือว่าจะให้มาระหว่างที่ cpu รันโปรแกรมครับ → ได้มาระหว่างการ run โปรแกรม

- ถ้าอย่างนั้นต้องกะให้ instruction ที่ใช้ค่า M,N ทำงานใน clock ที่มี M,N เข้ามาหรือครับ แต่ถ้าคำสั่งก่อนหน้านี้มันทำงานเสร็จก่อนหรือหลัง clock ที่มีค่า M,N เข้ามาต้องทำยังไงครับ (grader นับจังหวะไหนเพื่อรู้ว่าควรให้ค่า M,N มาครับ) → grader จะกำหนดค่า M, N นานๆ เพื่อให้ครับ

Comment 39 หลังจากที่ได้แสดงค่าหมดแล้ว ( done = 1 ) ถ้า

case 1 : มีการกด result เพื่อขอดู output อีกรอบ ให้ done = 0 แล้วคงค่า valid ใช้นิยามครับ → ใช้นิยาม เพราะถือว่าคำตอบถูกต้อง แต่ว่าการแสดงผลยังไม่เสร็จ

case 2 : กด start เพื่อให้ทำงานอีกรอบ ให้ done = 0 , valid = 0 ใช้นิยามครับ → ใช้นิยาม เพราะยังทำโปรแกรมอยู่

case 3 : กด reset ให้ done = 0 , valid = 0 ใช้นิยามครับ → ใช้นิยาม เพราะตอนนี้ล้างค่าทุกอย่าง ยังห้ามเอาคำตอบไปใช้

Comment 39 เวลา pram หมด ให้ terminate หรือ วนกลับไปเริ่ม 0 ใหม่คะ → ถ้ารันไปจนหมดถึง 0xFF แต่ยังไม่เจอคำสั่งให้ STOP จะไม่เกิดขึ้นครับ → test case ที่ทำจะต้องมี STOP ให้ → แต่ถ้าจะทำเผื่อก็ให้ไปวน 0x00 ก็ได้ครับ

Comment 40 การ reset ค่าใน rRAM ต้อง reset ทุกตัวหรือแค่ 16 ตัวแรกครับ → เพียง 16 ตัวแรกก็พอครับ

Comment 41 0 ยกกำลัง 0 เท่ากับเท่าไรครับ → 1

Comment 42 ถ้าต้องการเขียน ASM Chart โดยที่ข้อมูลรับ input ขณะอยู่ใน state นั้นๆแล้ว process ใน state นั้นๆเลย ควรเขียนยังงั้นหรือครับ เพราะถ้าเขียนเป็นแบบตัดสินใจแล้วไปยังสื่เปลี่ยนถัดไปมันจะนับเป็น state ใหม่ → ไป state ใหม่ แล้วเสียดาย clock ที่ผ่านไปหรืออย่างไร? → ใช้นิยามถ้าไป state ใหม่จะเกิด delay ขึ้นเลยต้องการให้วนรอ input นั้นๆ ระหว่างใน state เลยครับ → regC สามารถรับค่าได้จากทั้ง M, N ทำให้เราเดาไม่ได้ด้วยว่าค่าจะกำหนดให้อ่านจากไหน :( → ส่วนตัวใช้การรอ input ตรง progLoad ครับ เวลา progLoad มาแล้วรอไป state ใหม่แล้วค่อยอ่านค่ามัน



จะเข้าไป 1 clock ครบทำให้ค่าไม่ครบ เลยเลือกที่จะให้รับค่าระหว่างอยู่ใน state รอเลย แต่พอจะเขียน ASM chart ก็ตัดสินใจไม่ถูกว่าจะนำตัว decision ที่ตัดสินใจว่า progLoad เป็น 1 ไปไว้ตรงตำแหน่งใด ครบ → ค่า M และ N จะมาตอนรันโปรแกรมนะ -> ผมหมายถึงค่า progIN ครบ 555 ขอโทษที่ทำให้สับสน ครบ → คือตอนอ่านโปรแกรมเข้าตอนต้นไม่ต้องสนใจค่า M และ N ครบ

Comment 43 หลังจากที่เรา reset แล้ว ต้อง progLoad ทุกครั้งไหมครับ หรือ reset แล้ว สามารถ start ได้เลยโดยไม่ต้อง progLoad ครบ → ทำ start ต่อได้เลยครับ เพราะ reset ไม่ได้ล้าง pRAM แต่ต้องทำการกำหนด address ที่จะทำงานเริ่มต้นที่ 0 → รวมถึงล้างค่าใน rRAM ด้วยใช่ไหมครับ → ใช่ครับ

Comment 44 มีโอกาสที่กด reset ก่อนที่ done จะติดไหมครับ → มีครับ

Comment 45 ตอนทำ accA CMP accB ตรงที่เป็นจริงจะกลายเป็น 1 และที่เหลือที่แม้จะเป็น 0 หรือ 1 จากก่อนหน้า ก็จะกลายเป็น 0 ใช่ไหมครับ → ใช่ครับผลของ CMP จะมี 1 ได้อันเดียวที่เหลือ 0

Comment 46 ถ้าทำ CPU แบบ Multiple cycle ควรจะทำให้ได้กี่ CPI ครับ → ทำเท่าไรก็ได้ครับ ส่วนใหญ่ก็ 4-5 → ถ้าให้ดีสัก 4 หรือสัก 3.5

Comment 47 หลังจากส่งคำตอบเสร็จ 16 ตัวแล้วถ้า progLoad จะเป็น 1 reset ก่อนทุกครั้งมั้ยครับ หรือ ไม่ต้อง reset ก็ progLoad ได้เลยแล้วถ้าเป็นอย่างหลัง มันจะทำตัวเหมือน reset หรือเปล่าที่ valid กับ done ต้องกลับเป็น 0 → ใช่ครับไม่พร้อมส่งคำตอบ valid, done ต้องเป็น 0 ครบ → ใน test case ถ้าจะโหลดโปรแกรมใหม่ จะมีการส่งสัญญาณ reset ก่อนเสมอครับ

Comment 48 ถ้า fetch กับ write ที่มาจาก instruction เดียวกันไม่สามารถทำใน rising edge เดียวได้อันนี้ไม่นับว่าเป็น single clock cpu ใช่มั้ยครับ → ใช่ครับ

Comment 49 flag (eq, gt, ls) สามารถเป็น 1 พร้อมกันทั้งหมดได้ไหมคะ → ทั้งหมดไม่ได้

Comment 50 สามารถเปิดไฟล์ digital ของ cpu ที่เราต่อขณะ present ได้ไหมครับ → น่าจะได้นะครับ(ปีฟ้าได้นะ) → ทำได้และควรทำเลยครับ เพื่อนำเสนอพร้อม demo

Comment 51 เอกสาร นี้สามารถเลือกอย่างไร อย่างหนึ่งได้ไหมครับ ระหว่าง ASM Chart กับ FSM Chart ว่าจะใส่อันไหน → ก็ได้ครับ คือมี 10 หน้าให้ใช้

Comment 52 ตัว multi-cycle ที่เป็น 5 กลุ่มแรก จำเป็นต้องทำ Pipeline ไหมครับ หรือเป็น optional ครับ แล้วใน test case มีการรัน cpi ให้ไหมครับว่ามีให้อย่างน้อยกี่ cycle ถ้ามีจะมี ให้กี่ cycle ครับ → ไม่ต้อง pipeline ครับ ส่วน cpi คือไม่ต่ำกว่า 4 ครับ

Comment 53 Extra credit ที่ต้องทำ verilog ใช้ CPU ที่เป็น single clock ได้ไหมครับ → คือต้องทำขึ้นมาเองนะ ไม่ใช่ export จาก Digital

Comment 54 ใน test case 03\_P02T01 มี cycle ในการรันให้ประมาณ 26.5 cycles และมีคำสั่งที่ต้องรัน 15 คำสั่ง เป็นประมาณ 1.77 cpi แต่ใน comment 46 Multicycle แนะนำให้มีประมาณ 4/3.5 cpi ดังนั้นถ้าจะ

เอา extra ต้องทำแบบ pipeline อย่างเดียวเลยรีปแล้วครับ → ตอนปล่อย testcase ไม่ทันได้นับ clock ครับ  
ตอนนี้เพิ่มให้แล้ว

Test case 04\_P03T01 ก็เช่นกันครับ มี 11 คำสั่ง 25-26 Cycles ประมาณ 2.2 cpi ซึ่งก็ไม่ทันครับ → ปรับ  
เรียบร้อยครับ

Comment 55 presentation ใช้คอมตัวเองเปิดสไลด์ใช้ใหม่ครับ → ใช่ครับ

Comment 56 ในเทสเคสของอาจารย์ที่ปรับมาแล้ว 2 เคสสุดท้าย CPI ก็ยังน้อยอยู่ดีใหม่ครับ เช่น  
เคสสุดท้าย 08\_P06T02 มีคำสั่งให้ 13 instructions แต่ว่ามี clock ให้แค่ประมาณ 42 clock ถ้าคิด CPI ก็  
จะเป็น 3.23 ไม่ถึง 4/3.5 ใหม่ครับ → จัดเพิ่มให้เรียบร้อยครับ

Comment 57 ในกรณีที่ flag(eq, gt, ls) เป็น 1, 1, 0 ถ้าต่อมามีการเรียก opcode compare a กับ b แล้วได้  
ว่า a มากกว่า b จึ flag(eq, gt, ls) จะเป็น 0, 1, 0 แทนใช้รีปแล้วครับ → ใช่ครับ เรียก compare แล้วจะทำให้  
flag เหลือเป็น 1 ได้แค่ตัวเดียว

Comment 58 test case 8 start แรกมี 14 คำสั่งซึ่งควรจะมี 56 cycle แต่มีเวลารันให้ 42.5 cycle  
Test 7 มี 14 คำสั่งเหมือนกัน แต่มีเวลารันให้ 51.5 cycle → จัดเพิ่มให้เรียบร้อย

Comment 59 หาก CPU ของกลุ่มผมทำออกมาได้แค่ 2 CPI อันนี้ก็ยังถือว่ามีสิทธิ์ในการได้ Extra credit  
สำหรับการส่งผ่าน 5 กลุ่มแรกอยู่หรือไม่ครับ หรือว่าผมจำเป็นต้องทำให้ CPU มี CPI ที่มากขึ้นเป็น 3.5-4  
CPI แทน ถึงจะนับว่าเป็น Multi-Cycle ตาม Comment 46 → อันนี้คงแตก state เป็น 2 ส่วนออกมาใช้  
มั้ยครับ ก็อนุโลมได้ครับ

Comment 60 test case 8 start อันที่สอง cycle ก็ไม่พอครับมี cycle มาให้ 47.5 cycle → ปรับเรียบร้อย  
ครับ

Comment 61 Data Path สามารถทำในรูปแบบตารางได้มั้ยครับ → ปกติเป็นแนววงจร ทำตารางอาจจะดู  
แปลกๆนะ

Comment 62 futureให้ทำ ห.ร.ม หรือ ค.ร.น. หรือครับ เห็นว่าเป็น LCM ครับ → ครน ใช่ครับ

Comment 63 lcm(0, 0) มีค่าเป็นเท่าใดครับ → ไม่นิยามครับ ไปเริ่มที่ 1 เลย

ใครที่ทำแบบ Multi Cycle และส่งผ่าน Grader ใน Project Full Testcase ได้ครบทุก test case ก่อน ช่วย มาพิมพ์ชื่อกลุ่มไว้ด้วยนะครับ พร้อม capture หน้าจอที่แสดงว่าผ่านทุก test case

User	Problem	Language	Submit at	Result	Score	IP
<a href="#">(6733021521) นายศิริติ เศรษฐสุข</a>	<a href="#">[DigLo67_Project_Full]</a> DigLo67_Project_Full	Digital	2024-10-23 03:09:25	PPPPPPPPPPPPPP	100.0	49.237.9.195
<a href="#">(6733169821) นายพิรพล ศรีสุวรรณ</a>	<a href="#">[DigLo67_Project_Full]</a> DigLo67_Project_Full	Digital	2024-10-23 03:12:16	PPPPPPPPPPPPPP	100.0	2001:44c8:417b:9571:90b9:112b:ecaa:80a7
<a href="#">(6733285921) นายอนรรฆวี ชูจินดา</a>	<a href="#">[DigLo67_Project_Full]</a> DigLo67_Project_Full	Digital	2024-10-23 03:15:24	PPPPPPPPPPPPPP	100.0	161.200.190.130
<a href="#">(6733169821) นายพิรพล ศรีสุวรรณ</a>	<a href="#">[DigLo67_Project_Full]</a> DigLo67_Project_Full	Digital	2024-10-23 03:16:31	PPPPPPPPPPPPPP	100.0	2001:44c8:417b:9571:90b9:112b:ecaa:80a7
<a href="#">(6733089221) นายชนกฤต แสงปานแก้ว</a>	<a href="#">[DigLo67_Project_Full]</a> DigLo67_Project_Full	Digital	2024-10-23 03:20:34	PPPPPPPPPPPPPP	100.0	184.22.102.8
<a href="#">(6733170321) นายพิรพล อุดมฤกษ์</a>	<a href="#">[DigLo67_Project_Full]</a> DigLo67_Project_Full	Digital	2024-10-23 03:21:26	PPPPPPPPPPPPPP	100.0	1.47.142.211
<a href="#">(6733173221) นายพิรพล สุรินทร์วรางกูร</a>	<a href="#">[DigLo67_Project_Full]</a> DigLo67_Project_Full	Digital	2024-10-23 03:28:01	PPPPPPPPPPPPPP	100.0	2403:6200:88a0:82c7:e551:4ab:58e5:66f1
<a href="#">(6733211421) นายเมธาพันธ์ โพธิ์ลาครันนกุล</a>	<a href="#">[DigLo67_Project_Full]</a> DigLo67_Project_Full	Digital	2024-10-23 03:40:56	PPPPPPPPPPPPPP	100.0	2405:9800:b651:275a:ad89:b0c9:edea:2b18
<a href="#">(6733287121) นายอภิฏ เขียวบางยาง</a>	<a href="#">[DigLo67_Project_Full]</a> DigLo67_Project_Full	Digital	2024-10-23 03:42:30	PPPPPPPPPPPPPP	100.0	2001:fb1:a0:7bfa:f800:6c05:e49d:b6c6
<a href="#">(6733089221) นายชนกฤต แสงปานแก้ว</a>	<a href="#">[DigLo67_Project_Full]</a> DigLo67_Project_Full	Digital	2024-10-23 03:42:48	PPPPPPPPPPPPPP	100.0	184.22.102.8
<a href="#">(6733272521) นายณัฐวิทย์ ละอองศิริ</a>	<a href="#">[DigLo67_Project_Full]</a> DigLo67_Project_Full	Digital	2024-10-23 03:47:48	PPPPPPPPPPPPPP	100.0	124.120.116.2
<a href="#">(6733287121) นายอภิฏ เขียวบางยาง</a>	<a href="#">[DigLo67_Project_Full]</a> DigLo67_Project_Full	Digital	2024-10-23 03:47:49	PPPPPPPPPPPPPP	100.0	2001:fb1:a0:7bfa:f800:6c05:e49d:b6c6
<a href="#">(6733287121) นายอภิฏ เขียวบางยาง</a>	<a href="#">[DigLo67_Project_Full]</a> DigLo67_Project_Full	Digital	2024-10-23 03:50:11	PPPPPPPPPPPPPP	100.0	2001:fb1:a0:7bfa:f800:6c05:e49d:b6c6
<a href="#">(6733071921) นายณัฐวัฒน์ มาโนขุนทด</a>	<a href="#">[DigLo67_Project_Full]</a> DigLo67_Project_Full	Digital	2024-10-23 03:51:08	PPPPPPPPPPPPPP	100.0	2001:fb1:f9:f64b:e144:86d6:97c6:8f6e
<a href="#">(6733172621) นายพิรพล เพ็ญนิ้ง</a>	<a href="#">[DigLo67_Project_Full]</a> DigLo67_Project_Full	Digital	2024-10-23 03:51:11	PPPPPPPPPPPPPP	100.0	2001:44c8:41d0:385:7cb5:ff6d:615d:f890
<a href="#">(6733289421) นายอภิวิทย์ แสงอังคนาวิน</a>	<a href="#">[DigLo67_Project_Full]</a> DigLo67_Project_Full	Digital	2024-10-23 03:54:07	PPPPPPPPPPPPPP	100.0	27.55.95.141

# 1. FE 67

MAIN

Submissions

Hall of Fame

นายศิริติ เศรษฐสุข

Live submit

Write your code in the following box, choose language, and click submit button when finished

22

<elementAttributes>

23

<entry>

24

<string-Label/>string

25

<string-N/>string

26

</entry>

27

<entry>

28

<string-Bits/>string

29

<int>0/>int

30

</entry>

31

</elementAttributes>

32

<pos x="-240" y="240"/>

33

</visualElement>

34

<visualElement

35

<elementName-In/>elementName

36

<elementAttributes>

37

<entry>

38

<string-Label/>string

39

<string-progIn/>string

40

</entry>

41

<entry>

42

<string-Bits/>string

43

<int>1/>int

44

</entry>

45

</elementAttributes>

46

<pos x="240" y="300"/>

47

</visualElement>

48

<visualElement>

49

<elementName-In/>elementName

50

<elementAttributes>

51

<entry>

52

<string-Label/>string

53

<string-reset/>string

54

</entry>

55

</elementAttributes>

56

<pos x="240" y="360"/>

57

</visualElement>

Submission

Task

DigLo67\_Project\_Full

DigLo67\_Project\_Full

20.0 (s) | 512 (MB)

Description

[\[Read\]](#) [\[🔗\]](#)

Attachment

[\[File\]](#) [\[🔗\]](#)

Language

Digital

เลือกไฟล์

ไม่ได้เลือกไฟล์ใด

Submit

Latest Submission Status

Refresh

Sub ID

2045652

less than a minute ago (15:09:32)

100.0 [PPPPPPPPPPPPPP]

compiler msg

src

submissions (6)

testcases

2045652 2024-10-23 03:09:25  
OK

## 2. เสดถึ#1 ไฟ

The screenshot shows the CEDT Grader interface for problem 2045659. The main area contains a code editor with XML code for a circuit diagram. The right sidebar shows the submission details for 'DigLo67 Project Full'.

**Submission Details:**

- Task: DigLo67 Project Full
- Description: DigLo67 Project Full
- Attachment: [File]
- Language: Digital
- Submit button

**Latest Submission Status:**

- Sub ID: 2045659
- Time: less than a minute ago (15:12:23)
- Status: 100.0 [PPPPPPPPPPPPPPPP]
- Buttons: compiler msg, src, submissions (5), testcases

OK 2045659 2024-10-23 03:12:16

## 3. Minecraft Enjoyer

The screenshot shows the CEDT Grader interface for problem 2045667. The main area contains a code editor with XML code for a circuit diagram. The right sidebar shows the submission details for 'DigLo67 Project Full'.

**Submission Details:**

- Task: DigLo67 Project Full
- Description: DigLo67 Project Full
- Attachment: [File]
- Language: Digital
- Submit button

**Latest Submission Status:**

- Sub ID: 2045667
- Time: less than a minute ago (15:15:34)
- Status: 100.0 [PPPPPPPPPPPPPPPP]
- Buttons: compiler msg, src, submissions (6), testcases

2045667 2024-10-23 03:15:24

OK

#### 4 กลุ่ม SSS

MAINSubmissionsHall of Fame

Live submit

Write your code in the following box, choose language, and click submit button when finished

```
1 //xml version="1.0" encoding="utf-8"?>
2 <circuit>
3   <version /><version>
4   <attributes>
5     <visualElements>
6       <visualElement>
7         <elementName>Tunnel</elementName>
8         <elementAttributes>
9           <entry>
10             <string Name/><string>
11               <string clk/><string>
12                 <entry>
13                   <elementAttributes>
14                     <pos x="140" y="-1040"/>
15                   </visualElement>
16                 <visualElement>
17                   <elementName>Register</elementName>
18                   <elementAttributes>
19                     <entry>
20                       <string Label/><string>
21                         <string acc/><string>
22                           <entry>
23                             <string Bits/><string>
24                               <int s/><int>
25                               <entry>
26                                 <elementAttributes>
27                                   <pos x="140" y="-1040"/>
28                                 </visualElement>
29                               <visualElement>
30                                 <elementName>Register</elementName>
31                                 <elementAttributes>
32                                   <entry>
33                                     <string Label/><string>
34                                       <string acc/><string>
```

Submission

TaskDigLo67 Project Full  
DigLo67 Project Full  
20.0 (s) | 512 MB

Description[Read](#) [File](#)

Attachment[File](#) [File](#)

LanguageDigital

Choose FileNo file chosen

Submit

Latest Submission StatusRefresh

Sub ID2045688  
less than a minute ago (15:20:41)  
[C](#) 100.0 [PPPPPPPPPPPPPP]

[compiler msg](#) [src](#) [submissions \(3\)](#) [testcases](#)

OK

## 5 iMAKECPU

[illegible]

OK

## 6 ข่ายสี่มี code

[MAIN](#) [Submissions](#) [Hall of Fame](#) นายอภิภู เพ็ญบางยาง

Write your code in the following box, choose language, and click submit button when finished

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <circuit>
3   <version>2</version>
4   <attributes>
5     <entry>
6       <string-fromContent</string>
7       <romList>
8         <fromList>
9           </fromList>
10        </attributes>
11      </entry>
12    <visualElements>
13      <visualElement>
14        <elementName-In</elementName>
15        <elementAttributes>
16          <entry>
17            <string-Label</string>
18            <string-progID</string>
19          </entry>
20        </entry>
21        <string-Bits</string>
22        <int>15</int>
23      </elementAttributes>
24      <pos x="640" y="300"/>
25    </visualElement>
26    <visualElement>
27      <elementName-In</elementName>
28      <elementAttributes>
29        <entry>
30          <string-Label</string>
31          <string-progLoad</string>
32        </entry>
33      </entry>
34      <pos x="640" y="340"/>
35    </visualElement>
36  </circuit>
```

Submission

Task  
**DigLo67\_Project\_Full**  
DigLo67\_Project\_Full  
20.0 (s) | 512 (MB)

Description  
[\[Read\]](#)

Attachment  
[\[File\]](#)

Language  
Digital

Choose File No file chosen

Submit

Latest Submission Status Refresh

Sub ID 2045776  
less than a minute ago (15:42:36)  
100.0 [PPPPPPPPPPPPPPPP]

[compiler msg](#) [src](#) [submissions \(1\)](#) [testcases](#)

## 7 (สำรวจกรณีทีกลุ่ม 1 - 5) ไม่ได้ทำแบบ Multi Cycle Bit me up

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <circuit>
3   <version>2</version>
4   <attributes>
5     <visualElements>
6       <visualElement>
7         <elementName-In</elementName>
8         <elementAttributes>
9           <entry>
10            <string-Label</string>
11            <string-M</string>
12          </entry>
13        </entry>
14        <string-Bits</string>
15        <int>8</int>
16      </entry>
17      <entry>
18        <string-intFormat</string>
19        <intFormat-hex</intFormat>
20      </entry>
21    </elementAttributes>
22    <pos x="2540" y="1240"/>
23  </visualElement>
24  <visualElement>
25    <elementName-In</elementName>
26    <elementAttributes>
27      <entry>
28        <string-Label</string>
29        <string-N</string>
30      </entry>
31    </entry>
32    <string-Bits</string>
33    <int>8</int>
34  </entry>
35  </circuit>
```

Submission

Task  
**DigLo67\_Project\_Full**  
DigLo67\_Project\_Full  
20.0 (s) | 512 (MB)

Description  
[\[Read\]](#)

Attachment  
[\[File\]](#)

Language  
Digital

Choose File No file chosen

Submit

Latest Submission Status Refresh

Sub ID 2045807  
less than a minute ago (15:51:19)  
100.0 [PPPPPPPPPPPPPPPP]

[compiler msg](#) [src](#) [submissions \(9\)](#) [testcases](#)

## 8 (สำรวจกรณีทีกลุ่ม 1 - 5) ไม่ได้ทำแบบ Multi Cycle 13 bits

MAIN
Submissions
Hall of Fame
นายอภิวิชญ์ แสงอังกนาวิน

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <ircuit>
3   <version>2</version>
4   <attributes/>
5   <visualElements>
6     <visualElement>
7       <elementName>In</elementName>
8       <elementAttributes>
9         <entry>
10          <string>Label</string>
11          <string>M</string>
12        </entry>
13        <entry>
14          <string>Bits</string>
15          <int>8</int>
16        </entry>
17      </elementAttributes>
18      <pos x="1520" y="-380"/>
19    </visualElement>
20    <visualElement>
21      <elementName>In</elementName>
22      <elementAttributes>
23        <entry>
24          <string>Label</string>
25          <string>N</string>
26        </entry>
27        <entry>
28          <string>Bits</string>
29          <int>8</int>
30        </entry>
31      </elementAttributes>
32      <pos x="1520" y="-320"/>
33    </visualElement>
34    <visualElement>
35      <elementName>In</elementName>
36      <elementAttributes>

```

Submission

Task
**DigLo67\_Project\_Full**  
DigLo67\_Project\_Full  
20.0 (s) | 512 (MB)

Description
[\[Read\]](#)

Attachment
[\[File\]](#)

Language
Digital

Choose File
No file chosen

Submit

Latest Submission Status Refresh

Sub ID
2045818  
less than a minute ago (15:57:16)  
100.0 [PPPPPPPPPPPPPPPP]

[compiler msg](#)
[src](#)
[submissions \(5\)](#)
[testcases](#)

https://cedt-grader.nattee.net/submissions/get\_latest\_submission\_status/8858/1581

## 9 (สำรวจกรณีที่กลุ่ม 1 - 5) ไม่ได้ทำแบบ Multi Cycle กลุ่ม

MAIN
Submissions
Hall of Fame
นายเสาวฤทธิ์ สิงห์ทัศน์

Live submit

Write your code in the following box, choose language, and click submit button when finished

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <ircuit>
3   <version>2</version>
4   <attributes/>
5   <visualElements>
6     <visualElement>
7       <elementName>In</elementName>
8       <elementAttributes>
9         <entry>
10          <string>Label</string>
11          <string>M</string>
12        </entry>
13        <entry>
14          <string>Bits</string>
15          <int>8</int>
16        </entry>
17      </elementAttributes>
18      <pos x="1720" y="-1680"/>
19    </visualElement>
20    <visualElement>
21      <elementName>In</elementName>
22      <elementAttributes>
23        <entry>
24          <string>Label</string>
25          <string>M</string>
26        </entry>
27        <entry>
28          <string>Bits</string>
29          <int>8</int>
30        </entry>
31      </elementAttributes>
32      <pos x="1720" y="-1680"/>
33    </visualElement>
34    <visualElement>
35      <elementName>In</elementName>
36      <elementAttributes>

```

Submission

Task
**DigLo67\_Project\_Full**  
DigLo67\_Project\_Full  
20.0 (s) | 512 (MB)

Description
[\[Read\]](#)

Attachment
[\[File\]](#)

Language
Digital

Choose File
No file chosen

Submit

Latest Submission Status Refresh

Sub ID
2045893  
1 minute ago (16:29:00)  
100.0 [PPPPPPPPPPPPPPPP]

[compiler msg](#)
[src](#)
[submissions \(4\)](#)
[testcases](#)

## 10 (สำรวจกรณีที่กลุ่ม 1 - 5) ไม่ได้ทำแบบ Multi Cycle Dynamite programming

cedt-grader.nattee.net/submissions/2045898/edit

MAIN Submissions Hall of Fame

นายอวีน จุฬานนท์

Write your code in the following box, choose language, and click submit button when finished

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <circuit>
3   <version>2</version>
4   <attributes>
5     <entry>
6       <string>rowContent</string>
7       <romList>
8         <roms/>
9       </romList>
10    </entry>
11    <entry>
12      <string>showDataGraph</string>
13      <boolean>true</boolean>
14    </entry>
15    <entry>
16      <string>showDataTable</string>
17      <boolean>true</boolean>
18    </entry>
19  </attributes>
20  <visualElements>
21    <visualElement>
22      <elementName-In</elementName>
23      <elementAttributes>
24        <entry>
25          <string>label</string>
26          <string>M</string>
27        </entry>
28        <entry>
29          <string>Bits</string>
30          <int>8</int>
31        </entry>
32      </elementAttributes>
33      <pos x="100" y="100"/>
34    </visualElement>
35    <visualElement>
36      <elementName-In</elementName>
```

Submission

Task

DigLo67\_Project\_Full  
DigLo67\_Project\_Full  
20.0 (s) | 512 (MB)

Description

[Read](#) [File](#)

Attachment

[File](#) [Image](#)

Language

Digital

Choose File

No file chosen

Submit

Latest Submission Status

Refresh

Sub ID

2045898

less than a minute ago (16:30:31)

100.0 [PPPPPPPPPPPPPPPP]

compiler msg

src

submissions (5)

testcases

Search

16:31

23/10/2567