บทที่ 3
รหัส
CODES

วัตถุประสงค์

เมื่อนักศึกษา ศึกษาจบบทนี้แล้ว นักศึกษาสามารถ

1. ใส่รหัสให้กับเลขฐานสิบใด ๆ และถอดรหัสต่าง ๆ ให้เป็นเลขฐานสิบได้
2. อธิบายรหัส BCD รหัสแทนทั้งตัวเลข ตัวอักษร และสัญลักษณ์ได้
3. อธิบายรหัสชนิดที่มีน้ำหนัก ชนิดที่ไม่มีน้ำหนัก รหัสที่เป็นคอมพลีเมนต์ในตัวเอง รหัสซีเควนเชียลได้
4. แปลงระหว่างเลขฐานสองกับรหัสเกรย์ได้
5. อธิบายบิทแพริตี้ แพริตี้คำได้ และเติมแพริตี้ 2 ชนิดนี้เข้าไปในรหัสได้ นอกจากนี้ยังตรวจสอบข้อมูลผิดพลาดได้

การเข้ารหัสเป็นวิธีหนึ่งในการระบุลักษณะเฉพาะ เช่น จำนวนเลข ตัวอักษร หรือสัญลักษณ์ โดยใช้สัญลักษณ์อื่น เหตุผลอีกอย่างหนึ่งของการใช้รหัสก็เพื่อความปลอดภัย บุคคลอื่นไม่สามารถอ่านข่าวสารของเราได้ สำหรับในคอมพิวเตอร์นั้นรหัสถูกใช้ในการระบุลักษณะเฉพาะดังกล่าวข้างต้น โดยอาศัยสัญลักษณ์ 1 และ 0 ของเลขฐานสอง การเลือกใช้รหัสชนิดใดนั้นขึ้นอยู่กับหน้าที่หรือการใช้งานซึ่งรหัสจะเหมาะสม รหัสบางประเภทเหมาะแก่การดำเนินงานเลขคณิต บางประเภทดี เพราะมีประสิทธิภาพสูง กล่าวคือให้รายละเอียดได้มากกว่าโดยเพียงแต่ใช้จำนวนบิทสองสามบิทเท่านั้น ความสำคัญต่อไปคือรหัสซึ่งสามารถตรวจสอบความผิดพลาด หรือแก้ไขข้อผิดพลาดได้ อันนี้หมายความว่าเป็นรหัสซึ่งทำให้คอมพิวเตอร์สามารถพิจารณาได้ว่าลักษณะเฉพาะที่ถูกเข้ารหัสและส่งข้อมูลไปนั้น คอมพิวเตอร์ได้รับไปถูกต้องหรือไม่ ถ้ามีความผิดพลาดก็สามารถแก้ไขข้อผิดพลาดได้ด้วย

การคมนาคมของมนุษย์กับคอมพิวเตอร์ทำให้เกิดความจำเป็นในการใส่รหัส (encode) และถอดรหัส (decode) อย่างไรก็ตามเลขฐานสองซึ่งเป็นรหัสสำหรับจำนวนเลขฐานสิบ (หรือตัวอักษร หรือสัญลักษณ์) ที่เราคุ้นเคยนั้น ต้องใช้จำนวนบิทของเลขฐานสองมากกว่าเลขฐานสิบซึ่งมันแทนอยู่ ถ้าให้ N เป็นจำนวนเลขที่จะแทนด้วยเลขฐานสอง i บิท หรือด้วยเลขฐานสิบ j หลัก จะได้ความสัมพันธ์ว่า

$$N = 2^i = 10^j$$

$$i \log_{10} 2 = j$$

$$i = 3.32\, j$$

นั่นคือจำนวนบิทของเลขฐานสองจะเป็น 3.3 เท่าโดยประมาณของจำนวนหลักของเลขฐานสิบเมื่อแทนเลขจำนวนเดียวกัน สำหรับการแทนตัวอักษร และสัญลักษณ์ต่าง ๆ ต้องใช้รหัสซึ่งเป็นเลขฐานสอง 6 บิท, 8 บิท จำนวนบิทมาก ๆ ซึ่งต้องใช้ในการแสดงรายละเอียดเช่นนี้ยังความไม่สะดวกแก่มนุษย์ อุปกรณ์อินพุทสู่คอมพิวเตอร์ส่งข้อมูลเป็นตัวเลขตัวอักษรจึงต้องใส่รหัสในรูปของเลขฐานสอง เอาท์พุทจากคอมพิวเตอร์ก็ต้องถอดรหัสเพื่อแปลรหัสฐานสองให้เป็นตัวเลขตัวอักษรที่สามารถสื่อความหมายแก่มนุษย์ได้

เมื่ออินพุทสู่คอมพิวเตอร์เป็นเพียงตัวเลข เราอาจลดความไม่สะดวก อันเนื่องจากจำนวนบิทของเลขฐานสองมาก โดยใช้เลขฐานอื่นซึ่งเป็นกำลังของ 2 เช่น ฐานแปด (2^3) หรือ ฐานสิบหก (2^4)

3.1 รหัส BCD
BCD Codes

รหัส BCD (Binary – Coded Decimal) เป็นรหัสฐานสองที่ใส่รหัสแก่เลขฐานสิบแต่ละหลัก อาจใช้เลขฐานสองตั้งแต่ 4 บิทขึ้นไป เพื่อแทนตัวเลข 0 ถึง 9 ในเลขฐานสิบ ตัวอย่าง

รหัส BCD เช่นในตาราง 3.1

ตาราง 3.1 รหัส BCD

Decimal	8421	2421	5211	XS3
0	0000	0000	0000	0011
1	0001	0001	0001	0100
2	0010	0010	0011	0101
3	0011	0011	0101	0110
4	0100	0100	0111	0111
5	0101	1011	1000	1000
6	0110	1100	1010	1001
7	0111	1101	1100	1010
8	1000	1110	1110	1011
9	1001	1111	1111	1100

3.2 รหัสที่มีน้ำหนัก
Weighted Binary Codes

รหัสฐานสองที่มีน้ำหนัก เป็นรหัสที่ปฏิบัติตามหลักของน้ำหนักประจำตำแหน่ง แต่ละตำแหน่งของตัวเลขจะแทนน้ำหนักเฉพาะซึ่งบ่งบอกโดยชื่อของรหัส ตัวอย่างเช่น รหัส 8421 2421 5211 ในตาราง 3.1 ล้วนเป็นรหัสที่มีน้ำหนัก

รหัส 8421 เป็นรหัสฐานสองซึ่งใช้เลขฐานสองขนาด 4 บิท เพื่อแทนเลขฐานสิบทีละหลัก รหัส 8421 มีน้ำหนักตามลำดับของเลขฐานสอง คือ 8, 4, 2, 1 รหัส 8421 นี้บางครั้งจึงเรียกว่ารหัส BCD หมายความว่าถ้ากล่าวถึงรหัส BCD โดด ๆ ให้เข้าใจว่า หมายถึง รหัส 8421

ตัวอย่าง 3.1 จงใส่รหัส 8421 ให้แก่เลขฐานสิบ 428

วิธีทำ เลขฐานสิบ 4 2 8

รหัส 8421 0100 0010 1000

จากตาราง 3.1 และตัวอย่าง 3.1 เราสรุปความสัมพันธ์ของรหัส 8421 กับเลขฐานสองได้ว่ารหัส 8421 จะมีรูปเหมือนเลขฐานสองขนาด 4 บิทตั้งแต่เลข 0 ถึงเลข 9 ในเลขฐานสิบ เลขฐานสองขนาด 4 บิท อีก 6 ตัวที่เหลือคือ ตั้งแต่เลข 10 ถึง 15 ในเลขฐานสิบ จะไม่ปรากฏในรหัส 8421 ทั้งนี้เพราะการใส่รหัสฐานสอง BCD นั้น เป็นการใส่รหัสด้วยเลขฐานสอง ให้แก่เลขฐานสิบแต่ละหลักของจำนวนเลขฐานสิบทั้งจำนวน ดูตาราง 3.2 ประกอบ

ตาราง 3.2 ตารางเทียบค่าของเลขฐานสิบ, ฐานสอง และรหัส 8421

เลขฐานสิบ	เลขฐานสอง	รหัส 8421
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	0101
6	0110	0110
7	0111	0111
8	1000	1000
9	1001	1001
10	1010	0001 0000
11	1011	0001 0001
12	1100	0001 0010
13	1101	0001 0011
14	1110	0001 0100
15	1111	0001 0101
⋮		
98	1100010	1001 1000
100	1100100	0001 0000 0000
⋮		
2179	100010000011	0010 0001 0111 1001

คุณค่าสำคัญของรหัส BCD ก็คือ ความง่ายในการอ่านและระลึกถึงค่าทางตัวเลขที่รหัส BCD นั้นแทนอยู่ ดังจะเห็นได้จากตาราง 3.2 จำนวนเลขใหญ่ ๆ นั้น เลขฐานสองมีจำนวนบิทมาก ยากแก่การจดจำ แต่เมื่อแทนเป็นรหัสแล้วจะได้ กลุ่มเลขฐานสองกลุ่มละ 4 บิท ซึ่งง่ายและสะดวกต่อการหาค่า โดยเพียงการถอดรหัสเท่านั้น

ประโยชน์อย่างหนึ่งของรหัส 8421 เช่น ที่ฐานปฏิบัติการทดสอบจรวดนิวเคลียร์ ซึ่งแสงไฟจากรหัส BCD แสดงเวลาของวันสำหรับผู้สังเกตการณ์ที่อยู่ห่างออกไป 2 ไมล์ เวลาดังกล่าวคือ 6 : 32 : 4 จะถูกแทนด้วย

0000	0110	0011	0010	0100	0000
0	6	3	2	4	0

ตาราง 3.3 รหัส BCD ขนาด 4 บิท เพิ่มเติม

เลขฐานสิบ	3321	4221	5311	5421	6311	7421	$74\bar{2}\bar{1}$	$84\bar{2}\bar{1}$
0	0000	0000	0000	0000	0000	0000	0000	0000
1	0001	0001	0001	0001	0001	0001	0111	0111
2	0010	0010	0011	0010	0011	0010	0110	0110
3	0011	0011	0100	0011	0100	0011	0101	0101
4	0101	1000	0101	0100	0101	0100	0100	0100
5	1010	0111	1000	1000	0111	0101	1010	1011
6	1100	1100	1001	1001	1000	0110	1001	1010
7	1101	1101	1011	1010	1001	1000	1000	1001
8	1110	1110	1100	1011	1011	1001	1111	1000
9	1111	1111	1101	1100	1100	1010	1110	1111

3.2.1 รหัสที่เป็นคอมพลีเมนต์ในตัวเอง (Self–Complementing Codes)

บางตำราเรียกว่า รหัสสะท้อน (Reflective Codes) คือรหัสซึ่งการสลับที่กันระหว่าง 1 กับ 0 ในกลุ่มรหัส ซึ่งแทนเลขฐานสิบ d จะได้กลุ่มรหัสซึ่งแทนเลขฐานสิบ 9–d หรืออาจกล่าวได้ว่า 1's complement ของรหัสที่เป็นคอมพลีเมนต์ในตัวเอง ก็คือ 9's complement ของเลขฐานสิบนั้น ๆ รหัสประเภทนี้ได้แก่ รหัสเกิน 3 (XS3), 5211, 2421, 3321, 4221, $84\bar{2}\bar{1}$

ตัวอย่างเช่น กลุ่มรหัสสำหรับเลขฐานสิบ 0 คือ 0000 ของรหัส 2421 มี 1's complement คือ 1111 ซึ่งแทนเลขฐานสิบ 9 โดยที่ 9 เป็น 9's complement ของ 0 หรือรหัสเกิน 3 ของ 2_{10} (= d) คือ 0101 ถ้าสลับที่กันระหว่าง 0, 1 (หรืออาจกล่าวได้ว่าคือการหา 1's complement) จะได้ 1010 ซึ่งเป็นรหัสเกิน 3 ของ 7_{10} (= 9–d) โดยที่ 7_{10} คือ 9's complement ของ 2_{10}

3.2.2 รหัสซีเควนเชียล (Sequential Codes)

รหัสใด ๆ จะเรียกว่าเป็นรหัสซีเควนเชียล ก็ต่อเมื่อแต่ละรหัสของเลขฐานสิบที่สูงขึ้นเป็นเลขฐานสองที่มากกว่ารหัสของเลขฐานสิบก่อนหน้านั้นอยู่หนึ่ง สมบัติข้อนี้ช่วยได้มากต่อปฏิบัติการทางคณิตศาสตร์ของข้อมูล รหัสประเภทนี้ได้แก่ รหัส 8421, XS 3,

3.3 รหัสที่ไม่มีน้ำหนัก
Nonweighted Codes

เป็นรหัสซึ่งเลขฐานสองแต่ละบิทในกลุ่มรหัส ไม่มีค่าน้ำหนักประจำตำแหน่งของมัน

3.3.1 รหัสเกิน 3 (Excess–3 Code)

บางครั้งเรียกว่า XS3 เป็นรหัสซึ่งบวก 3 ให้กับแต่ละหลักของเลขฐานสิบที่ต้องการใส่รหัสก่อน แล้วจึงใช้เลขฐานสองขนาด 4 บิท แทนผลลัพธ์ที่ได้นั้น เนื่องจากเลขฐานสิบค่า

เล็กที่สุดคือ 0 เมื่อบวก 3 เข้าไปจะเป็น 3 ใช้เลขฐานสองแทนได้เป็น 0011 และเลขฐานสิบค่ามากที่สุดคือ 9 เมื่อบวก 3 เข้าไปจึงเป็น 12 ใช้เลขฐานสองแทนจะได้ 1100 ดังนั้นรหัสเกินสามของเลขฐานสิบตั้งแต่เลข 0 ถึง 9 จึงมีรูปเหมือนเลขฐานสองขนาด 4 บิท ตั้งแต่ 0011 จนถึง 1100 เลขฐานสองขนาด 4 บิทอีก 6 ตัวที่เหลือคือ 0000 0001 0010 1101 1110 1111 จะไม่ปรากฏอยู่ในรหัสเกิน 3

ตัวอย่าง 3.2 จงใส่รหัสเกิน 3 ให้กับเลขฐานสิบ 70492

วิธีทำ
$$7 + 3 = 10 = 1010$$
$$0 + 3 = 3 = 0011$$
$$4 + 3 = 7 = 0111$$
$$9 + 3 = 12 = 1100$$
$$2 + 3 = 5 = 0101$$

ดังนั้นรหัสเกิน 3 ของ 70492 คือ 1010 0011 0111 1100 0101

3.3.2 รหัสเกรย์ (Gray Codes)

เป็นรหัสที่ไม่มีน้ำหนัก มีสมบัติที่สำคัญคือ รหัสเกรย์ของจำนวนเลขที่อยู่ติดกันจะมีบิทต่างกันเพียง 1 บิทเท่านั้น รหัสเกรย์มีประโยชน์ใช้ในอุปกรณ์อินพุท-เอาท์พุท และอุปกรณ์ส่วนอื่น ๆ

ตาราง 3.4 รหัสเกรย์สำหรับเลขฐานสิบ 0-15

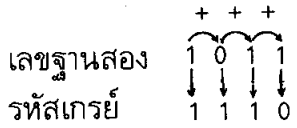
เลขฐานสิบ	เลขฐานสอง	รหัสเกรย์
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

การแปลงเลขฐานสองเป็นรหัสเกรย์

1. หลักแรกของรหัสเกรย์เหมือนกับบิทแรก (msb) ของเลขฐานสอง
2. บวกบิทถัดไป เพื่อให้ได้หลักต่อไปของรหัสเกรย์ โดยไม่คิดตัวทด

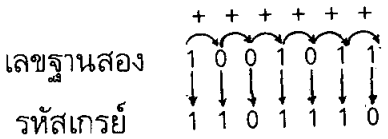
ตัวอย่าง 3.3 จงแปลงเลขฐานสอง 1011 ให้เป็นรหัสเกรย์

วิธีทำ



ตัวอย่าง 3.4 จงแปลงเลขฐานสอง 100 1011 ให้เป็นรหัสเกรย์

วิธีทำ

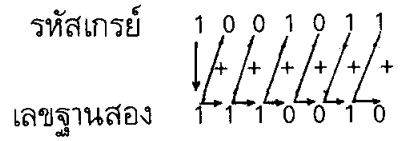


การแปลงรหัสเกรย์ให้เป็นเลขฐานสอง

1. บิทแรกของเลขฐานสอง (msb) เหมือนกับหลักแรกของรหัสเกรย์
2. ผลบวกของบิทล่าสุดของเลขฐานสองกับหลักถัดไปของรหัสเกรย์ โดยไม่คิดตัวทด คือเลขในบิทถัดไปของเลขฐานสอง

ตัวอย่าง 3.5 จงแปลงรหัสเกรย์ 1001011 ให้เป็นเลขฐานสอง

วิธีทำ



3.4 รหัสฐานแปด
Octal Code

รหัสฐานแปด ใช้เลขฐานสองขนาด 3 บิท ในการแทนเลขฐานแปดแต่ละหลัก ด้วยเหตุที่เลขฐานสองขนาด 3 บิท ตั้งแต่ 000 ถึง 111 เทียบค่ากับเลขฐานแปดจาก 0 ถึง 7 ซึ่งเป็นเลขพื้นฐานของระบบเลขฐานแปดได้พอดี ตัวอย่างเช่นต้องการใส่รหัสฐานแปดให้กับเลขฐานแปด 1724 จะได้

ข้อดีของรหัสฐานแปดคือ มันเป็นเลขฐานสองโดยธรรมชาติ ดังนั้นเราจึงอาจหาค่าเทียบเท่ากับเลขฐานสิบได้โดยใช้รหัสฐานแปด หรือเลขฐานแปดก็จะได้คำตอบเดียวกันเช่น 26_8 เขียนเป็นรหัสฐานแปดได้ว่า 010 110 ซึ่งสามารถอ่านเป็นเลขฐานสองได้โดยอัตโนมัติ

คือ 010110 จะให้ค่าเทียบเท่ากับเลขฐานสิบดังนี้

$$26_8 = 2 \times 8^1 + 6 \times 8^0 = 16 + 6 = 22_{10}$$

$$010110 = 1 \times 2^4 + 1 \times 2^2 + 1 \times 2^1 = 16 + 4 + 2 = 22_{10}$$

3.5 รหัสฐานสิบหก
Hexadecimal Codes

รหัสฐานสิบหกมีหลักการเช่นเดียวกับรหัสฐานแปด แต่ใช้เลขฐานสองขนาด 4 บิท ในการแทนตัวเลขในฐานสิบหก ตั้งแต่ 0 ถึง F

รหัสฐานสิบหกมีข้อดีเช่นเดียวกับรหัสฐานแปดในการเทียบค่าเลขฐานสิบได้โดยใช้เลขฐานสิบหกเอง หรือโดยรหัสฐานสิบหก เช่น 4E ในเลขฐานสิบหก คือ 0100 1110 ในรหัสฐานสิบหก จะเทียบค่าเลขฐานสิบได้ดังนี้

$$4E = 4 \times 16^1 + 14 \times 16^0 = 64 + 14 = 78_{10}$$

$$01001110 = 1 \times 2^6 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1$$

$$= 64 + 8 + 4 + 2 = 78_{10}$$

3.6 รหัส 5 บิท
5–Bit Codes

แม้ว่าการใส่รหัสให้แก่เลขฐานสิบจาก 0 ถึง 9 โดยใช้เลขฐานสองขนาด 4 บิท จะเพียงพอก็ตาม แต่ถ้าเพิ่มจำนวนบิทให้แก่รหัส ย่อมก่อให้เกิดความง่ายขึ้น หรือตรวจสอบความผิดพลาดได้ หรือแม้แต่แก้ไขข้อผิดพลาดได้ ตัวอย่างของรหัส 5 บิท บางชนิดแสดงอยู่ในตาราง 3.5

ตาราง 3.5 รหัส BCD ขนาด 5 บิท

Decimal	2–Out–of–5	51111	Shift–Counter
0	00011	00000	00000
1	00101	00001	00001
2	00110	00011	00011
3	01001	00111	00111
4	01010	01111	01111
5	01100	10000	11111
6	10001	11000	11110
7	10010	11100	11100
8	10100	11110	11000
9	11000	11111	10000

รหัส 2 จาก 5 (2–out–of–5) เป็นรหัสชนิดที่ไม่มีน้ำหนักมีสมบัติที่น่าสนใจคือ จะมีจำนวน 1 เพียง 2 ตัวในแต่ละกลุ่มรหัส จึงทำให้ง่ายต่อการตรวจสอบข้อมูลผิดพลาด รหัสชนิดนี้ใช้ในงานโทรศัพท์ และการสื่อสารต่าง ๆ

รหัส 51111 เป็นรหัสที่มีน้ำหนัก และเป็นรหัสที่เป็นคอมพลีเมนต์ในตัวเอง สังเกตว่ารหัส 51111 ของ 0 ถึง 4 จะมีเลข 1 ค่อย ๆ เพิ่มขึ้นจากบิทขวามือสุดไปทางซ้าย ในขณะที่รหัส 51111 ของ 5 ถึง 9 นั้น เลข 1 จะเพิ่มจำนวนขึ้นจากบิทซ้ายมือสุดไปทางขวา รหัสชนิดนี้จึงง่ายต่องานทางอิเล็กทรอนิกส์

รหัสชิฟท์เคาน์เตอร์ หรือ รหัสจอห์นสัน (shift–counter or Johnson code) เป็นรหัสไม่มีน้ำหนัก และง่ายต่องานทางอิเล็กทรอนิกส์เช่นเดียวกัน ด้วยรูปแบบของการจัดเรียงเลข 1 ของรหัสชนิดนี้

3.7 รหัสมากกว่า 5 บิท
More Than 5–Bit Codes

การเพิ่มจำนวนบิทให้แก่เลขฐานสองที่ใส่รหัสแก่เลขฐานสิบ จาก 0 ถึง 9 ทำให้ได้รหัสที่ง่ายต่อการตรวจสอบความผิดพลาดหรือง่ายต่อการดำเนินงานกับกลุ่มรหัส

ตาราง 3.6 ตัวอย่างรหัสมากกว่า 5 บิท

Decimal	Biquinary 50 43210	Ring–Counter 9876543210
0	01 00001	0000000001
1	01 00010	0000000010
2	01 00100	0000000100
3	01 01000	0000001000
4	01 10000	0000010000
5	10 00001	0000100000
6	10 00010	0001000000
7	10 00100	0010000000
8	10 01000	0100000000
9	10 10000	1000000000

รหัสไบควินารี่ (biquinary code) เป็นรหัส 7 บิท แบ่งเป็น 2 กลุ่มคือกลุ่มแรกมี 2 บิท เป็นตัวบอกว่ารหัสนั้น ๆ แทนเลข 0–4 หรือเลข 5–9 กลุ่มหลังมี 5 บิท โดยมี 1 อยู่เพียง 1 บิท ซึ่งจะเลื่อนจากบิทขวามือสุดไปทางซ้ายเรื่อย ๆ (ไบแปลว่าสอง, ควินารี่แปลว่าห้า) รหัสไบ-ควินารี่เป็นรหัสชนิดมีน้ำหนัก ตามบิทของมันเป็น 50 43210 การตรวจสอบข้อมูลผิดพลาดจะทำได้ง่าย เพราะแต่ละกลุ่มจะมี 1 เพียงตัวเดียว

รหัสริงเคาน์เตอร์ (ring–counter code) หรือรหัส 9876543210 เป็นรหัสที่ง่าย และสะดวกมากต่อการถอดรหัส รหัสนี้ใช้ 1 เพียงตัวเดียว ในรหัสแต่ละตัว

3.8 รหัสตรวจสอบความผิดพลาด
Error – Detecting Codes

ระบบดิจิตอลต้องมีความถูกต้องลึกถึงขั้นหลักต่าง ๆ ของตัวเลข ถ้าเกิดความผิดพลาดขึ้นจึงถือเป็นปัญหาร้ายแรง วิธีตรวจสอบข้อมูล โดยการสร้างรหัสชนิดต่าง ๆ ที่กล่าวมาแล้วในตอนก่อน ๆ เช่น รหัสไบควินารี่ รหัส 2 จาก 5 เป็นต้น เป็นหนทางหนึ่ง ในตอนนี้จะกล่าวถึงวิธีสำหรับตรวจสอบข้อมูลวิธีอื่น

บิทแพริตี้ (bit parity)

วิธีหนึ่งในการตรวจสอบข้อมูลผิดพลาดที่นิยมกันคือการใช้บิทแพริตี้ ความผิดพลาดทั้งหลายอาจมีสาเหตุมาจากสัญญาณรบกวน (noise) หรือสาเหตุอื่น ในเทปหรือดิสค์แม่เหล็ก (magnetic tape or disc) บางครั้งอาจมีแพริตี้เสริมเข้าในรหัสเพื่อเพิ่มความถูกต้องของการอ่านข้อมูลสู่คอมพิวเตอร์หรือในทางกลับกัน แพริตี้อาจเป็นได้ทั้งชนิดคี่ (odd) หรือชนิดคู่ (even) ซึ่งเมื่อเติมเข้าไปในรหัสแล้ว ทำให้จำนวนเลข 1 ในรหัสเป็นจำนวนคี่ หรือคู่ตามลำดับ การเติมบิทแพริตี้ก็ทำโดยใส่ 0 หรือ 1 เข้าไปในรหัสนั่นเอง ตัวอย่างเช่นการใส่แพริตี้เข้าไปยังรหัส 8421

ตาราง 3.7 รหัส 8421 พร้อมแพริตี้

Decimal	BCD Code	BCD with Odd Parity	BCD with Even Parity
0	0000	0000 1 or 00001	0000 0 or 00000
1	0001	0001 0 or 00010	0001 1 or 00011
2	0010	0010 0 or 00100	0010 1 or 00101
3	0011	0011 1 or 00111	0011 0 or 00110
4	0100	0100 0 or 01000	0100 1 or 01001
5	0101	0101 1 or 01011	0101 0 or 01010
6	0110	0110 1 or 01101	0110 0 or 01100
7	0111	0111 0 or 01110	0111 1 or 01111
8	1000	1000 0 or 10000	1000 1 or 10001
9	1001	1001 1 or 10011	1001 0 or 10010

ตัวอย่าง 3.6 จงตรวจดูว่าข้อมูลต่อไปนี้ถูกต้องหรือไม่

คำ	ประเภทแพริตี้
(ก) 1110111011 0	แพริตี้คี่
(ข) 1001010000 1	แพริตี้คู่
(ค) 1111010000 1	แพริตี้คู่

คำตอบ คำในข้อ (ก) ผิด คำในข้อ (ข), (ค) ถูก

3.9 รหัสแก้ไขความผิดพลาด
Error – Correcting Codes

แพริตี้คำ (word parity)

บิทแพริตี้สามารถตรวจสอบความผิดพลาดได้เพียงชนิดเดียวเท่านั้น ข้อมูลที่ผิดพลาดมากกว่า 1 บิทไม่อาจตรวจได้โดยบิทแพริตี้ การเติมแพริตี้คำเข้าไปในข้อมูลที่มีบิทแพริตี้อยู่จะไม่เพียงตรวจสอบความผิดพลาดได้เท่านั้น แต่ยังสามารถแก้ไขความผิดพลาดได้ด้วย ตัวอย่างในรูป 3.1 แสดงส่วนหนึ่งของเทปแม่เหล็ก ซึ่งบรรจุข้อมูลพร้อมด้วยแพริตี้ จุดขาวในรูปแทนค่า 0 และจุดดำแทนค่า 1 ข้อมูลทั้งหมดอาจเขียนเป็นเลขฐานสองได้ดังข้างล่างของรูป ในที่นี้ใช้แพริตี้ชนิดคี่

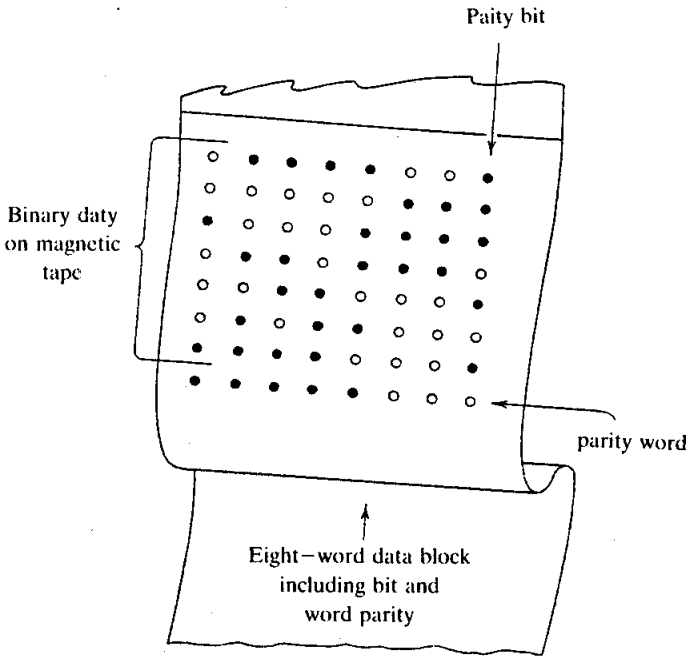


FIGURE 4–1 Section of magnetic tape showing word parity operation.

	Parity bit ↓
	01111001
	00000111
	10001111
	01101110
	00110001
	01011000
	11110001
Parity word →	11111000

รูป 3.1 ส่วนหนึ่งของเทปแม่เหล็กแสดงการทำงานของแพริตี้คำ

ถ้าเกิดความผิดพลาดขึ้นเช่นในแถวที่ 4 ข้อมูลที่ถูกต้องคือ [01] 101110 กลับกลายเป็น [10] 101110 เมื่อตรวจโดยบิทแพริตี้ยังคงได้จำนวนเลข 1 เป็นคี่ถูกต้อง แต่เมื่อตรวจโดยแพริตี้คำแล้วจะพบว่าคอลัมน์ซ้ายสุด 2 คอลัมน์เป็นแพริตี้ชนิดคู่ซึ่งผิดไป คอมพิวเตอร์จึงแก้ไขข้อมูลผิดพลาดนี้ได้

รหัสแฮมมิง (Hamming code)

บริษัทโทรศัพท์สนใจในการสื่อสารคมนาคมมากจึงสนับสนุนการวิจัยเพื่อหารหัสที่สามารถทั้งตรวจสอบความผิดพลาดและแก้ไขได้ด้วย แฮมมิ่งได้พัฒนาระบบที่ง่ายต่อการทำงานดังนี้ สมมุติว่าข้อมูลที่จะส่งประกอบด้วย 4 บิท รูปแบบของคำ ๆ นี้จะเป็น

$$D_7\ D_6\ D_5\ P_4\ D_3\ P_2\ P_1$$

โดย บิท D ทั้งหลายเป็นบิทข้อมูล และบิท P ทั้งหลายเป็นแพริตี้บิท

P_1 ถูกจัดขึ้นมาเพื่อให้เกิดแพริตี้คู่ในบิทชุด 1, 3, 5, 7 (P_1, D_3, D_5, D_7)

P_2 ถูกจัดขึ้นมาเพื่อให้เกิดแพริตี้คู่ในบิทชุด 2, 3, 6, 7 (P_2, D_3, D_6, D_7)

P_4 ถูกจัดขึ้นมาเพื่อให้เกิดแพริตี้คู่ในบิทชุด 4, 5, 6, 7 (P_4, D_5, D_6, D_7)

ลองดูตัวอย่างวิธีตรวจสอบและแก้ไขข้อมูลผิด ดังต่อไปนี้

ตัวอย่าง 3.7 ได้รับรหัสแฮมมิง 7 บิท เป็น 1111101 รหัสที่ถูกต้อง คืออะไร

วิธีทำ

D_7	D_6	D_5	P_4	D_3	P_2	P_1
1	1	1	1	1	0	1

บิท 4, 5, 6, 7 ไม่มีความผิดพลาด ⟶ 0 1 0

บิท 2, 3, 6, 7 ผิดพลาด ⟶ 0 1 0

บิท 1, 3, 5, 7 ไม่มีความผิดพลาด ⟶ 0 1 0

ได้ตัวเลข 0 1 0 หมายความว่า บิท 2 ผิดไป

ดังนั้นข้อมูลที่ถูกต้องคือ 1 1 1 1 1 1 1 **ตอบ**

ที่กล่าวมาแล้วเป็นรหัสแฮมมิง 7 บิท อย่างไรก็ตามหลักการนี้สามารถขยายไปสู่จำนวนบิทเท่าใดก็ได้ ตัวอย่างเช่นรหัส 15 บิท มีรูปแบบดังนี้

$$D_{15}\ D_{14}\ D_{13}\ D_{12}\ D_{11}\ D_{10}\ D_9\ P_8\ D_7\ D_6\ D_5\ P_4\ D_3\ P_2\ P_1$$

มีข้อสังเกตคือ แพริตี้บิทจะใส่เข้าไปที่แต่ละ 2^n บิท

3.10 รหัสแทนตัวเลขตัวอักษร
Alphanumeric Codes

ในคอมพิวเตอร์นอกจากจะใช้ตัวเลขแล้ว ยังมีตัวอักษร และสัญลักษณ์ต่าง ๆ รหัสต่อไปนี้เป็นรหัสที่ใช้แทนลักษณะเฉพาะเหล่านี้

รหัสฐานแปด (octal code)

ตัวอย่างของรหัสฐานแปดที่ใช้แทนตัวเลข ตัวอักษร สัญลักษณ์แสดงอยู่ในตาราง 3.8 เลขฐานแปด 2 หลักใช้แทนแต่ละลักษณะเฉพาะเหล่านี้ เลขฐานแปดตั้งแต่ 00 ถึง 77 สามารถแทนตัวเลข ตัวอักษร สัญลักษณ์ได้ 64 ตัว รหัสฐานแปดนี้เทียบเป็นเลขฐานสองได้เป็น 6 บิท (2 ชุด ๆ ละ 3 บิท) รหัสชนิดนี้ใช้ในเครื่องพิมพ์ดีดไฟฟ้าที่ต่อเข้ากับคอมพิวเตอร์

ถ้าเราใช้เครื่องพิมพ์ดีดเพื่อป้อนข้อมูลให้แก่อินพุทของคอมพิวเตอร์ แต่ละครั้งที่เรากดคีย์ (key) จะให้เลขฐานแปด 2 หลักหรือเลขฐานสอง 6 หลัก ส่งไปสู่คอมพิวเตอร์ในรูปพลังงานไฟฟ้า

ตาราง 3.8 รหัสฐานแปดแทนตัวเลขตัวอักษร

Character	6-Bit Code (in octal)	Character	6-Bit Code (in octal)
0	00	tab	40
1	01	N	41
2	02	O	42
3	03	P	43
4	04	Q	44
5	05	R	45
6	06	S	46
7	07	T	47
8	10	U	50
9	11	V	51
-	12	W	52
;	13	X	53
/	14	Y	54
!	15	Z	55
'	16	.	56
=	17	skip line	57
space	20	)	60
A	21	±	61
B	22	@	62
C	23	#	63
D	24	$	64
E	25	%	65
F	26	¢	66
G	27	&	67
H	30	*	70
I	31	(	71
J	32	–	72
K	33	:	73
L	34	?	74
M	35	°	75
,	36	"	76
carriage return	37	+	77

ตัวอย่างเช่น เมื่อพิมพ์ข้อความว่า BEGIN TEST จะปรากฏเป็นข้อมูลฐานแปดต่อไปนี้ป้อนสู่คอมพิวเตอร์

B	E	G	I	N		T	E	S	T	●
22	25	27	31	41	20	47	25	46	47	56

ในลักษณะเช่นนี้คอมพิวเตอร์ จะเก็บเลขฐานสอง (แทนเลขฐานแปดดังแสดงข้างบนนี้) เพื่อแทนข้อมูลตัวเลข ตัวอักษรในคอมพิวเตอร์

รหัส ASCII อินพุท/เอาท์พุท (ASCII input/output code)

รหัสมาตรฐานซึ่งยอมรับและใช้กันอย่างกว้างขวางมากในวงการอุตสาหกรรมคือ รหัส ASCII (ออกเสียงอ่านว่า as–kee) ย่อมาจาก American Standard Code for Information Interchange รหัสชนิดนี้พัฒนาขึ้นจากความต้องการรหัสฐานสองที่เป็นระเบียบสำหรับใช้กับตัวเลข ตัวอักษร และสัญลักษณ์ดังแสดงในตาราง 3.9 รหัส ASCII ใช้กันแพร่หลายหลายสำหรับเครื่องพิมพ์ (printers) และเทอร์มินัล (terminal) ซึ่งเชื่อมโยง (interface) กับระบบคอมพิวเตอร์ขนาดเล็กหรือแม้แต่ในระบบขนาดใหญ่

โดยพื้นฐานแล้วรหัส ASCII จะมี 7 บิท สำหรับบิทที่แปดอาจเติมเข้าไปแล้วจัดให้บิทนี้มีค่าเป็น 1 เสมอ หรือเป็น 0 เสมอ หรือให้เป็นแพริตี้บิท

ตาราง 3.9 รหัส ASCII

		Most Significant Digit (Hex)							
		0	1	2	3	4	5	6	7
	0	NUL	DLE	SP	0	@	P	`	p
	1	SOH	DC1	!	1	A	Q	a	q
	2	STX	DC2	"	2	B	R	b	r
	3	ETX	DC3	#	3	C	S	c	s
	4	EOT	DC4	$	4	D	T	d	t
	5	ENQ	NAK	%	5	E	U	e	u
Least	6	ACK	SYN	&	6	F	V	f	v
Significant	7	BEL	ETB	'	7	G	W	g	w
Digit	8	BS	CAN	(	8	H	X	h	x
(LSD)	9	HT	EM	)	9	I	Y	i	y
	A	LF	SUB	*	:	J	Z	j	z
	B	VT	ESC	+	;	K	[	k	{
	C	FF	FS	,	<	L	\	l	¦
	D	CR	GS	-	=	M	]	m	}
	E	SO	RS	.	>	N	^	n	~
	F	SI	US	/	?	O	_	o	DEL

ACK	Acknowledge	FF	Form feed
BEL	Bell	FS	File Separator
BS	Backspace	GS	Group Separator
CAN	Cancel	HT	Horizontal tabulation
CR	Carriage return	LF	Line feed
DC1	Device control 1	NAK	Negative acknowledge
DC2	Device control 2	NUL	Null or all zeros
DC3	Device control 3	RS	Record Separator
DC4	Device control 4	SI	Shift in
DEL	Delete	SO	Shift out
DLE	Data link escape	SOH	Start of heading
EM	End of medium	SP	Space
ENQ	Enquiry	STX	Start of text
EOT	End of transmission	SUB	Substitute
ESC	Escape	SYN	Synchronous idle
ETB	End of transmission block	US	Unit Separator
ETX	End of text	VT	Vertical tabulation

รหัส EBCDIC (EBCDIC Code)

รหัส EBCDIC (อ่านออกเสียงว่า eb–si–dik) ย่อมาจาก Extended Binary Coded Decimal Interchange Code เป็นรหัสซึ่งใช้มากในคอมพิวเตอร์ขนาดใหญ่ทั้งหลาย รหัส EBCDIC ไม่เหมือนรหัส ASCII ซึ่งใช้ลำดับของเลขฐานสองโดยตรงสำหรับแทนลักษณะเฉพาะต่าง ๆ แต่รหัส EBCDIC ใช้ BCD เป็นพื้นฐานของการจัดเลขฐานสอง

ตาราง 3.10 รหัส EBCDIC

		Most Significant Digit (Hex)															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0	NUL	DLE	DS		SP	&							{	}	\	0
	1	SOH	DC1	SOS				/		a	j	~		A	J		1
	2	STX	DC2	FS	SYN					b	k	s		B	K	S	2
	3	ETX	DC3							c	l	t		C	L	T	3
	4	PF	RES	BYP	PN					d	m	u		D	M	U	4
	5	HT	NL	LF	RS					e	n	v		E	N	V	5
	6	LC	BS	EOB	UC					f	o	w		F	O	W	6
LSD	7	DEL	IL	PRE	EOT					g	p	x		G	P	X	7
(Hex)	8		CAN							h	q	y		H	Q	Y	8
	9		EM							i	r	z		I	R	Z	9
	A	SMM	CC	SM		¢	!	¦	:								
	B	VT				.	$	,	#								
	C	FF	IFS		DC4	<	*	%	@								
	D	CR	IGS	ENQ	NAK	(	)	–	'								
	E	SO	IRS	ACK		+	;	>	=								
	F	SI	IUS	BEL	SUB	|	¬	?	"								

ตัวอย่าง 3.8 จงแทนเลขฐานสิบ 9 ในรูปแบบเลขฐาน และรหัสต่าง ๆ

คำตอบ เลขฐานสิบ 9

เลขฐานสอง 1001

เลขฐานแปด 11

เลขฐานสิบหก 9

รหัส ASCII 0111001

รหัส EBCDIC 11111001

รหัส BCD 1001

รหัส XS3 1100

รหัสเกรย์ 1101

รหัสไบควินนารี 10 10000

สรุป

รหัสฐานสองแบบต่าง ๆ ที่นิยมใช้กันอยู่ กลุ่มหนึ่งคือ รหัส BCD ซึ่งใช้แทนเลขฐานสิบตั้งแต่ 0 ถึง 9 อีกกลุ่มหนึ่งใช้แทนตัวเลข ตัวอักษร และสัญลักษณ์ ในรูปของคำเลขฐานสอง

รหัส BCD ที่นิยมใช้กันมากที่สุดคือ รหัส 8421 (รหัสนี้โดยทั่วไปเรียกกันง่าย ๆ ว่า รหัส BCD) เพราะเป็นรหัสที่แทนเลขฐานสิบจาก 0 ถึง 9 ด้วยค่าทางเลขฐานสองของ 0000 ถึง 1001

รหัส BCD มีทั้งชนิดที่มีน้ำหนัก และไม่มีน้ำหนัก มีจำนวนบิทในแต่ละกลุ่มรหัสแตกต่างกัน เช่น 4 บิท 5 บิท 7 บิท 10 บิท

รหัสฐานแปดหนึ่ง ๆ แทนด้วยเลขฐานสองขนาด 3 บิท ในขณะที่รหัสฐานสิบหกแทนด้วยเลขฐานสองขนาด 4 บิท จำนวนที่แสดงในรหัส 2 ชนิดนี้ เป็นเลขฐานสองโดยธรรมชาติแท้จริง ทำให้เป็นรหัสที่ใช้ในปฏิบัติการทางคณิตศาสตร์ได้ง่าย

เราอาจเสริมแพริตี้บิทเข้าไปในรหัสต่าง ๆ เพื่อประโยชน์ในการตรวจสอบความผิดพลาดของข้อมูล และอาจเสริมแพริตี้คำเข้าไปอีกเพื่อประโยชน์ในการแก้ไขความผิดพลาด นอกจากนี้ยังมีการคิดค้นรหัสชนิดที่สามารถตรวจสอบ และแก้ไขความผิดพลาดได้ในตัวเอง อย่างเช่น รหัสแฮมมิง

รหัส BCD อื่น ๆ นอกเหนือจากรหัส 8421 ก็เป็นที่นิยมใช้ เพราะมีลักษณะพิเศษที่อุปกรณ์อิเล็กทรอนิกส์สามารถจัดการได้สะดวก

แบบฝึกหัด

3.1 จงใส่รหัส 8421 BCD ให้กับเลขฐานสิบต่อไปนี้

(ก) 428 (ข) 1493 (ค) 76915

3.2 จงถอดรหัส 8421 BCD ต่อไปนี้ให้เป็นเลขฐานสิบ

(ก) 1001 0101 0001 1000

(ข) 0001 1010 0101 0111

3.3 จงใส่รหัสเกิน 3 ให้กับเลขฐานสิบต่อไปนี้

(ก) 125 (ข) 7493 (ค) 6012

3.4 จงถอดรหัสเกิน 3 ต่อไปนี้

(ก) 1010 0011 1000

(ข) 0111 1100 1001 0100

3.5 จงแปลงรหัสเกิน 3 ในข้อ 3.4 ให้เป็นรหัส $74\overline{21}$ และรหัส 8421

3.6 จงแปลงเลขฐานสิบต่อไปนี้เป็นรหัสเกรย์

(ก) 14 (ข) 27

3.7 จงแปลงรหัสเกรย์ต่อไปนี้เป็นเลขฐานสอง

(ก) 10101110 (ข) 1111101

3.8 คำต่อไปนี้ใช้แพริตี้คู่ จงตรวจสอบดูว่าคำใดผิด

(ก) 10101010 (ข) 11110100

(ค) 10010010 (ง) 10001000

3.9 จงเสริมแพริตี้ชนิดคู่ให้กับข้อมูลข้างล่างนี้

100110

010100

010111

111100

101010

101111

3.10 จงตรวจสอบดูว่าบิทใดผิดจากข้อมูลข้างล่างนี้ ซึ่งใช้แพริตี้คู่

แพริตี้บิท

101110

010111

100110

101101

010111

110101 ← แพริตี้คำ

3.11 จงแปลงเลขฐานสิบต่อไปนี้เป็นรหัส 2- จาก -5 และรหัสไบควินนารี

(ก) 276 (ข) 7658

3.12 จงใส่รหัสแฮมมิง 7 บิทใช้แพริตี้คู่ แก่เลขฐานสอง : 0101

3.13 จงตรวจสอบและแก้ไขความผิดพลาดของคำที่เป็นรหัสแฮมมิง 7 บิทใช้แพริตี้คู่ :

0101110

3.14 คำของรหัส ASCII ต่อไปนี้คืออะไร เมื่อเขียนอยู่ในรูปเลขฐานสิบหก :

D4 C9 D0 AD D4 CF D0

3.15 คำของรหัส EBCDIC อยู่ในรูปเลขฐานสิบหกดังข้างล่างนี้ คำนี้คืออะไร

C6 C9 D9 E2 E3 40 C3 C1 D9 C4