

1.3 โครงสร้างข้อมูลอาร์เรย์ (Array Structure)

อาร์เรย์ แบ่งตามลักษณะการจัดเก็บออกเป็น 2 ประเภทใหญ่ ๆ คือ

1. อาร์เรย์ขนาด 1 มิติ
2. อาร์เรย์ขนาดหลาย (N) มิติ

โดยโครงสร้างอาร์เรย์จะมีแนวคิดในการจัดเก็บดังนี้

- อาร์เรย์ 1 มิติ เกิดจากการนำตัวแปรเดี่ยวชนิดเดียวกัน ขนาดเดียวกันมา เรียงต่อๆ กัน
- อาร์เรย์ 2 มิติ เกิดจากการนำตัวแปรอาร์เรย์ 1 มิติมาต่อเรียงกัน
- อาร์เรย์ N มิติ เกิดจากการนำตัวแปรอาร์เรย์ N-1 มิติมาต่อเรียงกัน

องค์ประกอบของโครงสร้างข้อมูลแบบอาร์เรย์

จะประกอบด้วย

- ชื่อของอาร์เรย์
- มิติของอาร์เรย์
- ขนาดแต่ละช่องของอาร์เรย์
- ขอบเขตของอาร์เรย์ (มีค่าเป็นเลขบวกหรือลบก็ได้)
 - ขอบเขตด้านต่ำ (Lower Bound : l)
 - ขอบเขตด้านสูง (Upper Bound : u)

การเก็บอาร์เรย์ในหน่วยความจำ

ด้วยเหตุที่โครงสร้างของหน่วยความจำจะเป็นแบบมิติเดียว มีลักษณะคล้ายกับชั้น เก็บของหลายๆ ชั้นเรียงต่อๆ กัน โดยแต่ละชั้นสามารถเก็บข้อมูลได้ 1 ไบต์ หรือ 8 บิต เท่านั้น ดังนั้นการนำโครงสร้างของอาร์เรย์ที่มีขนาดของแต่ละช่องอาจมากกว่า 1 ไบต์ รวมทั้งยังลักษณะของมิติเข้ามาเกี่ยวข้องอีกด้วย จึงจำเป็นต้องทำการคำนวณเพื่อให้สามารถระบุตำแหน่งที่เก็บจริงในหน่วยความจำกับตำแหน่งที่เป็นมิติที่มนุษย์มอง ให้สอดคล้องกันให้ได้ เพื่อให้การจัดเก็บข้อมูลไปตามจุดประสงค์ของการใช้งาน อีกทั้งอาร์เรย์ยังเป็นโครงสร้างข้อมูล พื้นฐานของโครงสร้างอื่น ๆ อีก

ในภาษาระดับสูงต่างๆ มักจะมีโครงสร้างอาร์เรย์ให้ใช้งานอยู่แล้ว ผู้เขียนโปรแกรม ไม่จำเป็นต้องรู้จักวิธีการจัดเก็บจริงในหน่วยความจำ แต่อาร์เรย์ในทางวิศวกรรมคอมพิวเตอร์ จำเป็นต้องจะเจาะลึกถึงการสร้างโครงสร้างอาร์เรย์ในหน่วยความจำจริง ๆ เพื่อนำไปใช้กับ ภาษาระดับต่ำที่ไม่มี

โครงสร้างอาร์เรย์ใช้ หรือใช้ในงานไมโครคอนโทรลเลอร์และระบบฝังตัว (Embedded System) อีกทั้ง เพื่อเป็นพื้นฐานในการเขียนงานในเชิงลึกต่อไป

1.3.1 อาร์เรย์ขนาด 1 มิติ (One Dimension Array)

โครงสร้างอาร์เรย์ขนาด 1 มิติ จะมีลักษณะทางสถาปัตยกรรมเหมือนกับโครงสร้าง ของ หน่วยความจำ ดังนั้นจึงสามารถเปรียบเทียบลักษณะกันได้โดยง่าย แต่จะมีสิ่งที่แตกต่างกันอยู่อย่าง หนึ่งคือ ขนาดของช่องอาร์เรย์จะไม่ใช่ 1 ไบต์เสมอไป จะขึ้นอยู่กับชนิดและขนาด ของข้อมูล เช่น

ชนิดข้อมูล	ขนาด (Byte)
Short Integer	1
Integer	2
Long Integer	4
Byte	1
Word	2
Double Word	4
Single Float	4
Double Float	8
Character	1
Text	1-255
Boolean	1

รูปที่ 1.11 แสดงขนาดของข้อมูลแต่ละชนิด

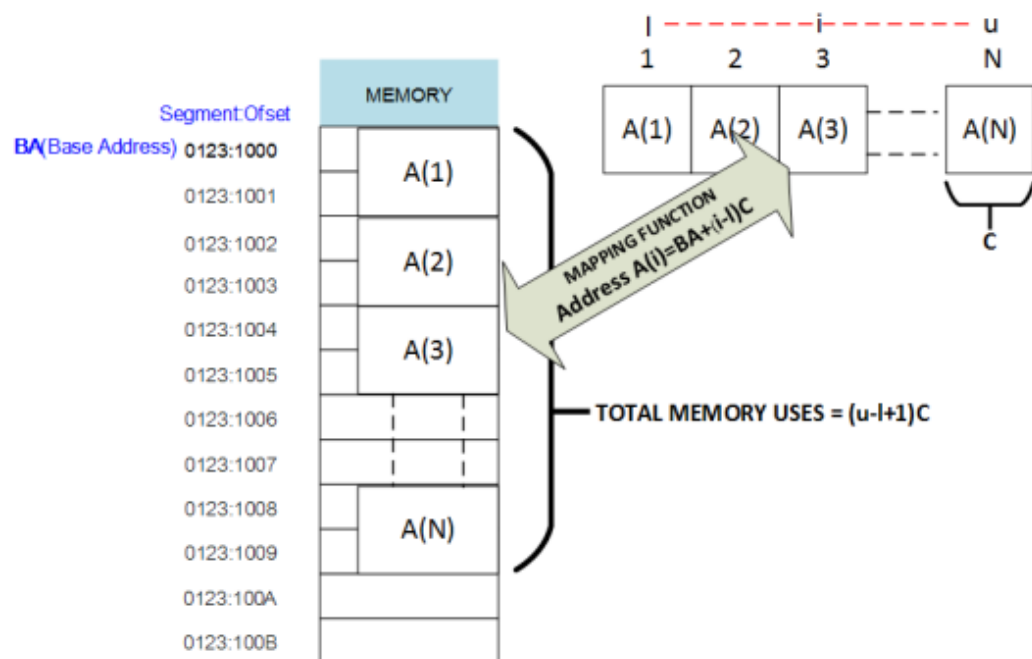
ทั้งนี้ก็จะขึ้นอยู่กับการออกแบบตัวแปลภาษา (Compiler) ว่าผู้ออกแบบจะกำหนด เท่าไร เพื่อความสะดวกใช้งาน ดังนั้นการเลือกใช้ชนิด และขนาดของข้อมูล จะต้องคำนึงถึง ความเหมาะสม ในการใช้งานเพื่อจุดประสงค์ดังต่อไปนี้

1. เพื่อใช้พื้นที่ของสื่อเก็บข้อมูลอย่างประหยัด และพอเพียง
2. เพื่อลดภาระการประมวลผลของซีพียู

การสร้างโครงสร้างอาร์เรย์ 1 มิติ

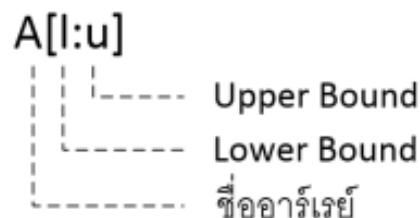
ก่อนใช้โครงสร้างอาร์เรย์ ต้องทำการสร้างโครงสร้างขึ้นมาก่อน โดยมีขั้นตอนดังนี้

1. คำนวณพื้นที่ทั้งหมดที่ต้องใช้
2. จองใช้พื้นที่
3. คำนวณหาตำแหน่งที่เก็บ
4. นำข้อมูลเข้าไปเก็บ หรืออ่านข้อมูลมาใช้งาน



รูปที่ 1.12 แสดงการนำอาร์เรย์ขนาด 1 มิติไปเก็บในหน่วยความจำ

รูปแบบการระบุโครงสร้างอาร์เรย์ขนาด 1 มิติ คือ $A[l:u]$ ดังนี้



และ สามารถสร้างสมการในการคำนวณค่าต่างๆ ได้ดังนี้

$$\begin{aligned} \text{จำนวนช่องทั้งหมด (Element)} &= (\text{ขอบเขตด้านสูง}-\text{ขอบเขตด้านต่ำ})+1 \\ &= (u-l+1) \end{aligned}$$

$$\begin{aligned} \text{จำนวนพื้นที่ทั้งหมด (Size)} &= (\text{จำนวนช่องทั้งหมด}) (\text{จำนวนไบต์ต่อช่อง}) \\ &= (u-l+1)C \end{aligned}$$

ตำแหน่งที่เก็บ (Address) = (ตำแหน่งเริ่มต้น)+(จำนวนช่องที่ข้าม)(จำนวน
ไบต์ต่อช่อง)

$$= BA+(i-l)C$$

หมายเหตุ u : Upper Bound

l : Lower Bound

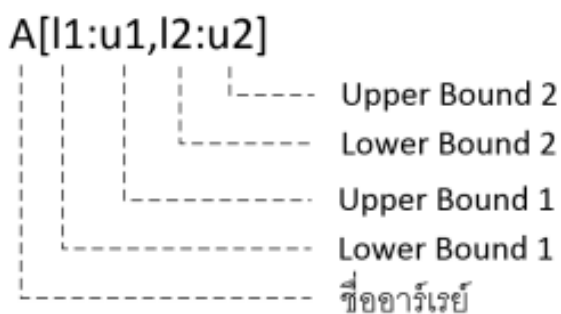
C : จำนวนไบต์ต่อช่อง

i : ซับสคริปต์ของอาร์เรย์

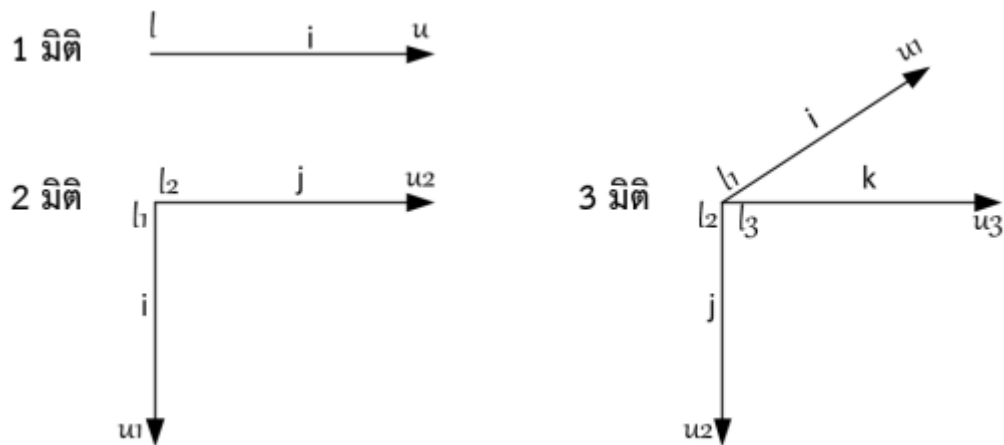
BA(Base Address) : ตำแหน่งเริ่มต้นของพื้นที่หน่วยความจำ

1.3.2 อาร์เรย์ขนาด 2 มิติ (Two Dimension Array)

อาร์เรย์ขนาด 2 มิติ จะมีรากฐานมาจากการนำอาร์เรย์ขนาด 1 มิติ มาต่อเรียงกันตาม หลักการ
ที่กล่าวไว้ข้างต้น การระบุโครงสร้างของอาร์เรย์จะมีส่วนของมิติเข้ามาเกี่ยวข้องดังนี้



การระบุตัวแปรประจำแกนของแต่ละมิติ ต้องระบุให้เป็นมาตรฐานตามหลักการทาง
คณิตศาสตร์ โดยใช้อักษร i, j, k,... แทนแกนของแต่ละมิติเป็นลำดับไปเรื่อย ๆ โดยถ้ามีเพียง มิติเดียว
จะระบุเป็นแกน โดยรูปของอาร์เรย์จะอยู่ในแนวตั้ง หรือแนวนอนก็ได้ ถ้าอาร์เรย์มี ตั้งแต่ 2 มิติขึ้นไป
การระบุตำแหน่งแกนต้องคำนึงถึงความเหมาะสมด้านโครงสร้าง และการ สื่อความหมายดังรูป 1.13



รูปที่ 1.13 แสดงการใช้ตัวแปร i j k ในการระบุชื่อแกนของแต่ละมิติ

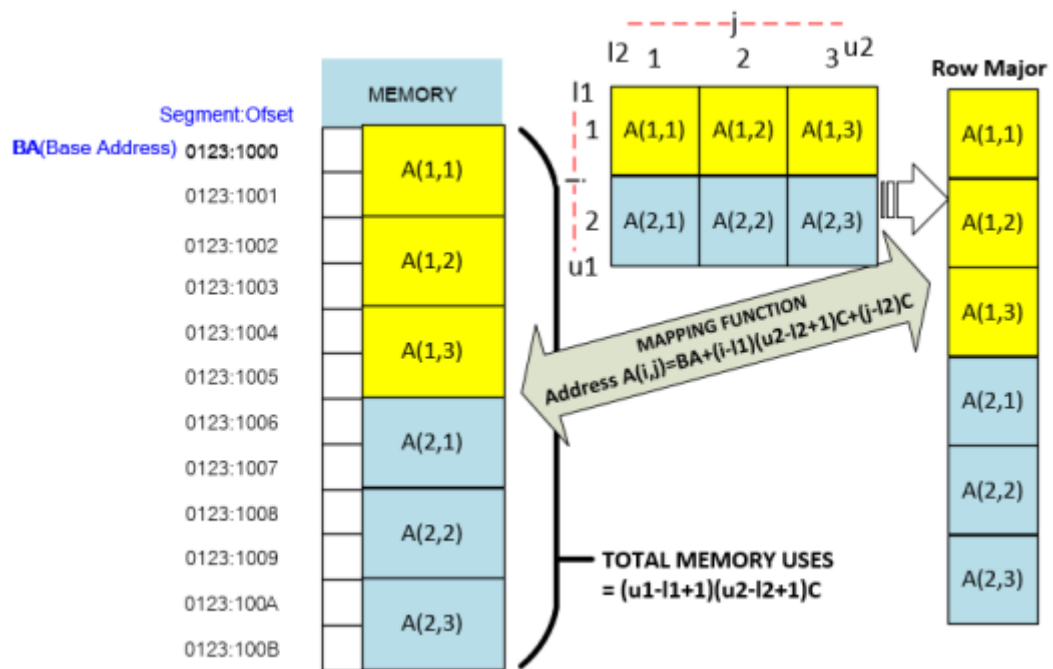
1.3.2.1 การตัดอาเรย์ขนาด 2 มิติ

ด้วยเหตุที่โครงสร้างของหน่วยความจำเป็นแบบ 1 มิติตามที่กล่าวไว้ข้างต้น ดังนั้นไม่ว่าอาเรย์จะเป็นกี่มิติก็ตาม ต้องทำการตัดและจัดให้เหลือเพียง 1 มิติให้ได้ เพื่อจะได้มีโครงสร้างที่สอดคล้องกับโครงสร้างหน่วยความจำ และเป็นแนวทางในการสร้างสมการเพื่อคำนวณค่าต่างๆ ต่อไป

อาเรย์ขนาด 2 มิติ สามารถตัดได้ 2 แบบ (หรือ 2! แบบ) คือ

1. ตัดตามแนวแถว (Row Major)
2. ตัดตามแนวสดมภ์ (Column Major)

การตัดตามแนวแถว (Row Major) : สมมติว่าอาเรย์มีขนาดเป็น $A[1:2,1:3]$ การตัดตามแนวแถวเพื่อให้เหลือเพียงภาพมิติเดียว ก็จะได้ตามรูป 1.14



รูปที่ 1.14 แสดงการติดตามแนวแถว (Row Major)

การหาสมการในการคำนวณค่าต่าง ๆ ก็มีพื้นฐานมาจากอาเรย์ขนาด 1 มิติ โดยมีหลักการดังนี้

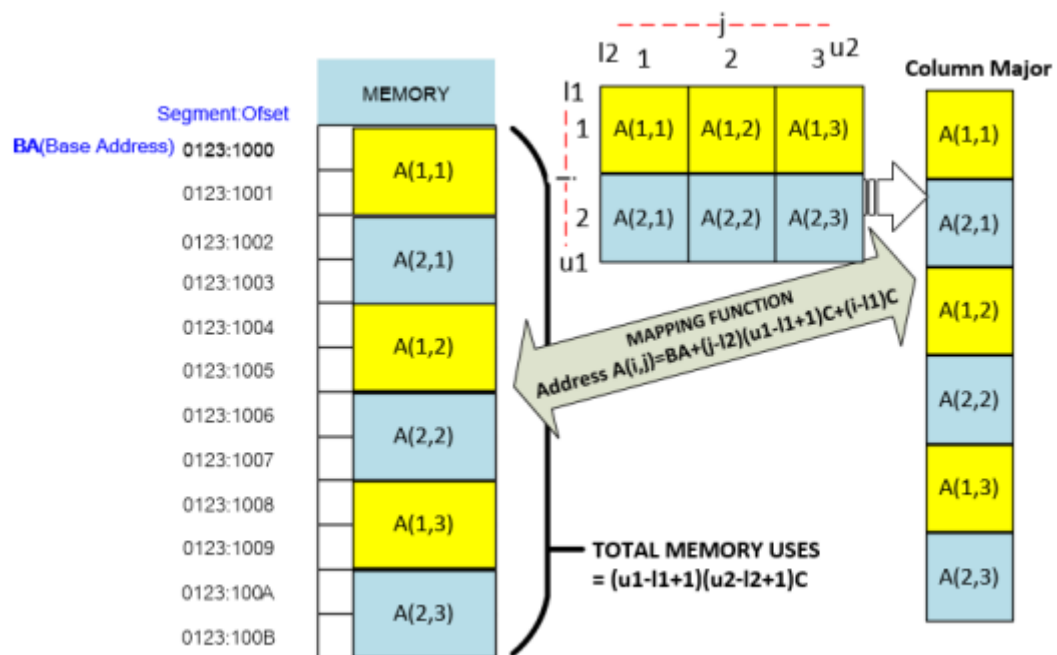
จำนวนช่องทั้งหมด (Element) เกิดจากการนำจำนวนช่องด้านมิติที่ 1 คูณกับจำนวนช่องด้านมิติที่ 2 จะได้สมการเป็น

$$\begin{aligned} \text{Element} &= (\text{จำนวนช่องของมิติ 1}) (\text{จำนวนช่องของมิติ 2}) \\ &= (u_1-l_1+1)(u_2-l_2+1) \\ \text{Size} &= (\text{Element})(\text{จำนวนไบต์ต่อช่อง}) \\ &= (u_1-l_1+1)(u_2-l_2+1)C \end{aligned}$$

ตำแหน่งที่อยู่ก็เริ่มคิดจากตำแหน่งอาเรย์ที่ต้องการคำนวณ (i, j) นั้นอยู่แถวที่เท่าไร ซึ่งการข้ามไปยังแถวนั้น ๆ ต้องข้ามไปเท่ากับจำนวนช่องของแต่ละแถว โดยในแกนแถวจะมีตัวแปร i กำกับอยู่ ดังนั้นการคำนวณก็ใช้ประโยชน์จากตัวแปรประจำแถว (คือตัวแปร i) โดยการนำค่า i กับจุดเริ่มต้นของแถว (l_1) มาหาส่วนต่างกัน ก็จะได้จำนวนแถวที่ข้าม แล้วก็นำมาคูณกับจำนวนช่องต่อแถวดังกล่าวข้างต้น ซึ่งจะได้คำตอบว่าอยู่แถวไหน พอทราบว่าจะอยู่แถวไหนแล้วที่เหลือก็จะเหมือนกับการคำนวณของแบบ 1 มิติ จากนั้นก็นำมาบวกจากแอดเดรสเริ่มต้นของหน่วยความจำที่จองใช้งาน (BA) ซึ่งก็จะได้สมการดังนี้

$$\begin{aligned}
 \text{Address} &= (\text{ตำแหน่งเริ่มต้น}) + (\text{จำนวนแถวที่ข้าม})(\text{จำนวนช่องแถว})(\text{จำนวนไบต์ต่อช่อง}) \\
 &\quad + (\text{จำนวนสดมภ์ที่ข้าม})(\text{จำนวนไบต์ต่อสดมภ์}) \\
 &= BA + (i-l_1)(u_2-l_2+1)C + (j-l_2)C
 \end{aligned}$$

การตัดตามแนวสดมภ์ (Column Major) : สมมติว่าอาเรย์มีขนาดเป็น $A[1:2,1:3]$ การตัดตามแนวสดมภ์เพื่อให้เหลือเพียงภาพมิติเดียว ก็จะได้ตามรูป 1.15



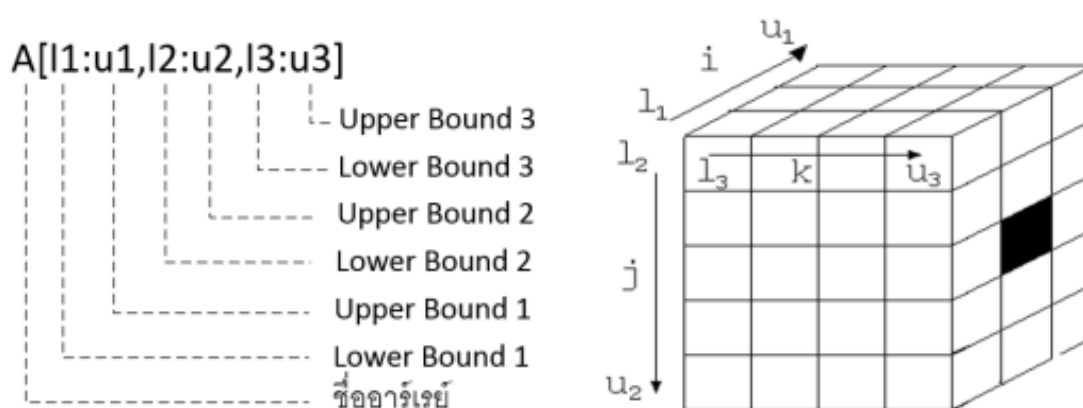
รูปที่ 1.15 แสดงการตัดตามแนวสดมภ์ (Column Major)

การหาสมการต่าง ๆ ก็จะคล้ายกับการตัดตามแนวแถว (Row Major) โดยสมการหาจำนวนช่อง และจำนวนพื้นที่จะเหมือนกัน จะแตกต่างเฉพาะสมการหาตำแหน่งที่อยู่ของอาเรย์เท่านั้นแต่หลักการคิดก็คล้ายกัน กล่าวคือ ตำแหน่งที่อยู่ก็เริ่มคิดจากตำแหน่งอาเรย์ที่ ต้องการคำนวณ (i,j) นั้น อยู่สดมภ์ที่เท่าไร ซึ่งการข้ามไปยังสดมภ์นั้น ๆ ต้องข้ามไปเท่ากับจำนวนช่องของแต่ละสดมภ์ โดยในแกนสดมภ์จะมีตัวแปร j กำกับอยู่ ดังนั้นการคำนวณก็ใช้ ประโยชน์จากตัวแปรประจำสดมภ์ (คือตัวแปร j) โดยการนำค่า j กับจุดเริ่มต้นของสดมภ์ (l_2) มาหาส่วนต่างกัน ก็จะได้จำนวนสดมภ์ที่ข้าม แล้วก็นำมาคูณกับจำนวนช่องต่อสดมภ์ ดังกล่าวข้างต้น ซึ่งจะได้คำตอบว่าอยู่สดมภ์ไหน พอทราบว่าจะอยู่สดมภ์ไหนแล้วที่เหลือก็จะเหมือนกับการคำนวณของแบบ 1 มิติ จากนั้นก็นำมาบวกจากแอดเดรสเริ่มต้นของ หน่วยความจำที่จองใช้งาน (BA) ซึ่งก็จะได้สมการดังนี้

$$\begin{aligned}
 \text{Address} &= (\text{ตำแหน่งเริ่มต้น}) + (\text{จำนวนสดมภ์ที่ข้าม})(\text{จำนวนช่องสดมภ์})(\text{จำนวนไบต์ต่อช่อง}) \\
 &\quad + (\text{จำนวนแถวที่ข้าม})(\text{จำนวนไบต์ต่อแถว}) \\
 &= BA + (j-l_2)(u_1-l_1+1)C + (i-l_1)C
 \end{aligned}$$

1.3.3 อาร์เรย์ขนาด 3 มิติ (Three Dimension Array)

อาร์เรย์ขนาด 3 มิติ จะมีรากฐานมาจากการนำอาร์เรย์ขนาด 2 มิติ มาต่อเรียงกัน การระบุโครงสร้างของอาร์เรย์จะมีส่วนของมิติเข้ามาเกี่ยวข้องดังนี้



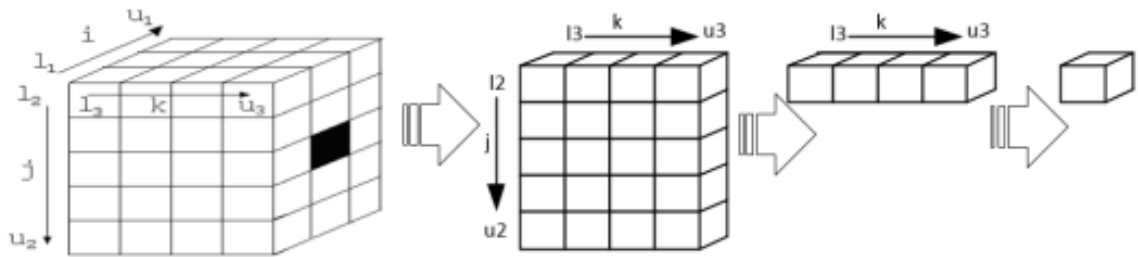
รูปที่ 1.16 แสดงโครงสร้างและแกนต่าง ๆ ของอาร์เรย์ 3 มิติ

1.3.3.1 การตัดอาร์เรย์ขนาด 3 มิติ

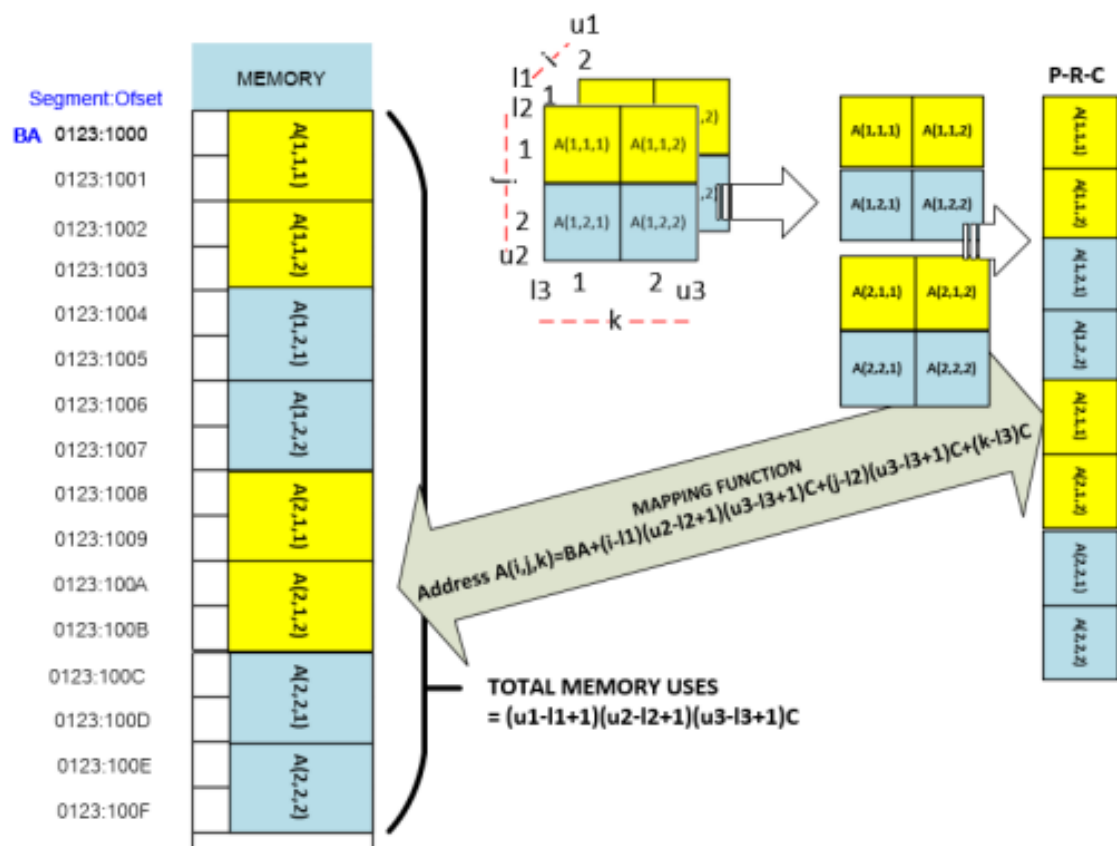
อาร์เรย์ขนาด 3 มิติ สามารถตัดได้ 6 แบบ (หรือ 3! แบบ) คือ

1. ระนาบ-แถว-สดมภ์ (Plane-Row-Column) หรือ Plane-Row Major
2. ระนาบ-สดมภ์-แถว (Plane-Column-Row) หรือ Plane-Column Major
3. แถว-ระนาบ-สดมภ์ (Row-Plane-Column) หรือ Row-Plane Major
4. แถว-สดมภ์-ระนาบ (Row-Column-Plane) หรือ Row-Column Major
5. สดมภ์-ระนาบ-แถว (Column-Plane-Row) หรือ Column-Plane Major
6. สดมภ์-แถว-ระนาบ (Column-Row-Plane) หรือ Column-Row Major

1.การตัดแบบ “ระนาบ-แถว-สดมภ์” (Plane-Row-Column) การตัดตามแนว ระนาบ-แถว-สดมภ์ เพื่อให้เหลือเพียงภาพมิติเดียว จะได้ตามรูป



รูปที่ 1.7 แสดงการตัดตามแนวระนาบ-แถว-สดมภ์



รูปที่ 1.18 แสดงความสัมพันธ์การตัดตามแนวระนาบ-แถว-สดมภ์ กับการเก็บใน หน่วยความจำ

การหาสมการในการคำนวณค่าต่าง ๆ ก็มีพื้นฐานมาจากอาเรย์ขนาด 2 มิติ โดยมี หลักการ ดังนี้

จำนวนช่องทั้งหมด (Element) เกิดจากการนำจำนวนช่องด้านมิติที่ 1 คูณกับ จำนวนช่องด้านมิติที่ 2 คูณกับจำนวนช่องด้านมิติที่ 3 จะได้สมการเป็น

$$\begin{aligned} \text{Element} &= (\text{จำนวนช่องของมิติ 1})(\text{จำนวนช่องของมิติ 2})(\text{จำนวนช่องของมิติ 3}) \\ &= (u_1-l_1+1)(u_2-l_2+1)(u_3-l_3+1) \end{aligned}$$

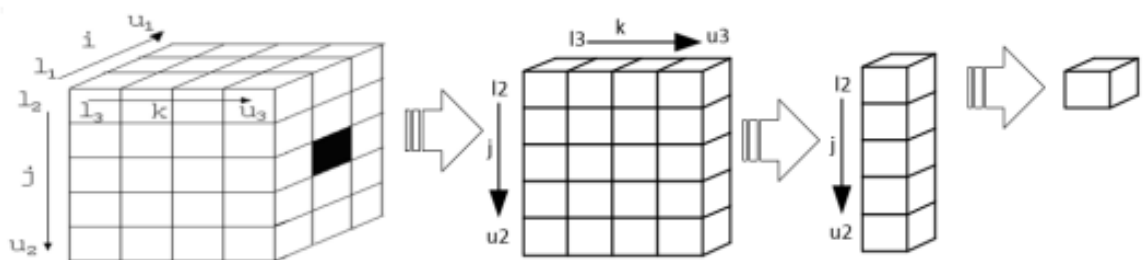
Size = (Element)(จำนวนไบต์ต่อช่อง)

$$= (u_1-l_1+1)(u_2-l_2+1)(u_3-l_3+1)C$$

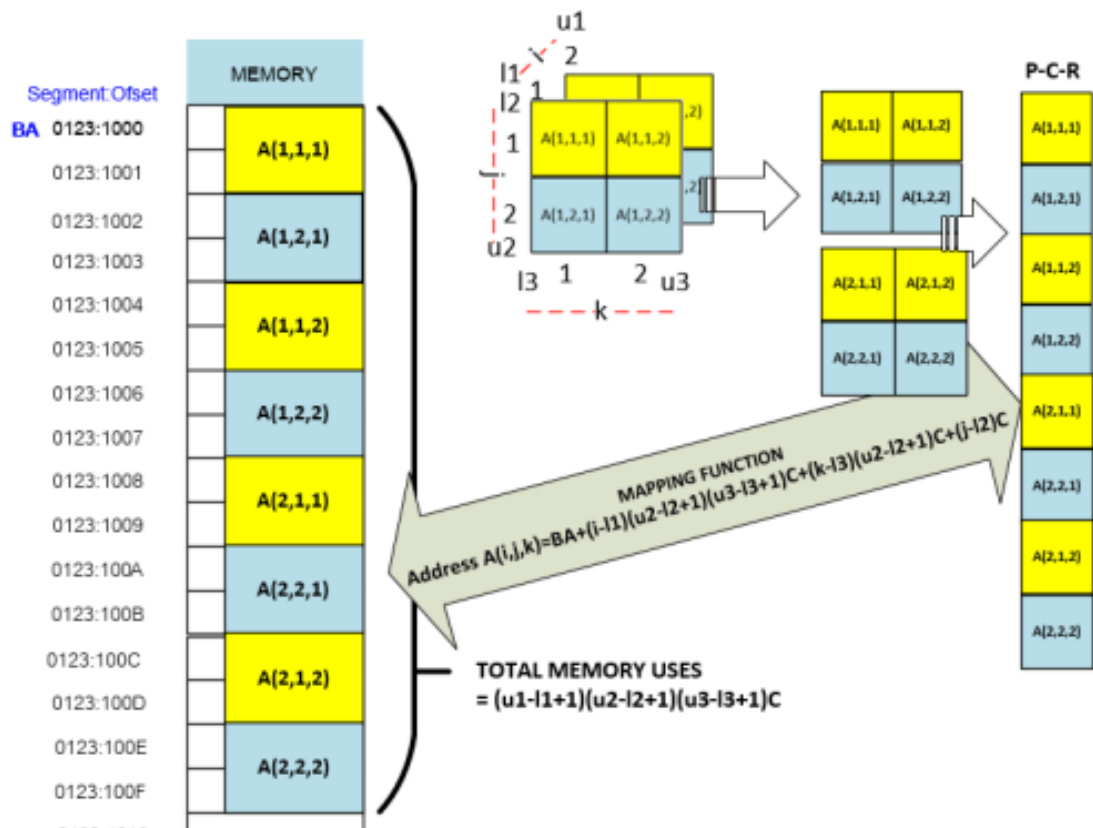
ตำแหน่งที่อยู่ก็เริ่มคิดจากตำแหน่งอาเรย์ที่ต้องการคำนวณ (i,j,k) นั้นอยู่ระนาบที่เท่าไร ซึ่งการข้ามไปยังระนาบนั้น ๆ ต้องข้ามไปเท่ากับจำนวนช่องของแต่ละระนาบ โดยใน แกนระนาบจะมีตัวแปร i กำกับอยู่ ดังนั้นการคำนวณก็ใช้ประโยชน์จากตัวแปรประจำระนาบ (ตัวแปร i) โดยการนำค่า i กับจุดเริ่มต้นของระนาบ (l_1) มาหาส่วนต่างกัน ก็จะได้จำนวน ระนาบที่ข้าม แล้วก็นำมาคูณกับจำนวนช่องต่อระนาบดังกล่าวข้างต้น ซึ่งจะได้คำตอบว่าอยู่ ระนาบไหน พอทราบว่ายู่ระนาบไหนแล้วที่เหลือก็จะเหมือนกับการคำนวณของแบบ 2 มิติ จากนั้นก็นำมาบวกจากแอดเดรสเริ่มต้นของหน่วยความจำที่จองใช้งาน (BA) ซึ่งก็จะได้ สมการดังนี้

$$\begin{aligned} \text{Address} &= (\text{ตำแหน่งเริ่มต้น}) + (\text{จำนวนระนาบที่ข้าม})(\text{จำนวนช่องต่อระนาบ})C + \\ &\quad (\text{จำนวนแถวที่ข้าม})(\text{จำนวนช่องต่อแถว})C + (\text{จำนวนสดมภ์ที่ข้าม})C \\ &= BA + (i-l_1)(u_2-l_2+1)(u_3-l_3+1)C + (j-l_2)(u_3-l_3+1)C + (k-l_3)C \end{aligned}$$

2.การตัดแบบ “ระนาบ-สดมภ์-แถว” (Plane-Column-Row) การตัดตามแนว ระนาบ-สดมภ์-แถว เพื่อให้เหลือเพียงภาพมิติเดียว จะได้ตามรูป 1.19



รูปที่ 1.19 แสดงการตัดตามแนวระนาบ-สดมภ์-แถว



รูปที่ 1.20 แสดงความสัมพันธ์การตัดตามแนวระนาบ-สตมภ์-แถว กับการเก็บในหน่วยความจำ

ตำแหน่งที่อยู่ก็เริ่มคิดจากตำแหน่งอาเรย์ที่ต้องการคำนวณ (i,j,k) นั้นอยู่ระนาบที่เท่าไร ซึ่งการข้ามไปยังระนาบนั้น ๆ ต้องข้ามไปเท่ากับจำนวนช่องของแต่ละระนาบ โดยใน แกนระนาบจะมีตัวแปร i กำกับอยู่ ดังนั้นการคำนวณก็ใช้ประโยชน์จากตัวแปรประจำระนาบ (ตัวแปร i) โดยการนำค่า i กับจุดเริ่มต้นของระนาบ ($l1$) มาหาส่วนต่างกัน ก็จะได้จำนวนระนาบที่ข้าม แล้วก็นำมาคูณกับจำนวนช่องต่อระนาบดังกล่าวข้างต้น ซึ่งจะได้คำตอบว่าอยู่ ระนาบไหน พอทราบว่าจะอยู่ระนาบไหนแล้วที่เหลือก็จะเหมือนกับการคำนวณของแบบ 2 มิติ จากนั้นก็นำมาบวกจากแอดเดรสเริ่มต้นของหน่วยความจำที่จองใช้งาน (BA) ซึ่งก็ได้ สมการดังนี้

$$\begin{aligned} \text{Address} &= (\text{ตำแหน่งเริ่มต้น}) + (\text{จำนวนระนาบที่ข้าม})(\text{จำนวนช่องต่อระนาบ})C + \\ &\quad (\text{จำนวนสตมภ์ที่ข้าม})(\text{จำนวนช่องต่อสตมภ์})C + (\text{จำนวนแถวที่ข้าม})C \\ &= BA + (i-l1)(u2-l2+1)(u3-l3+1)C + (k-l3)(u2-l2+1)C + (j-l2)C \end{aligned}$$

อนึ่งการคิดสมการของการแบบการตัดอื่น ๆ หากเข้าใจวิธีการตัดแบบ 2 แบบดัง ตัวอย่างแล้ว ก็จะสามารถหาสมการแบบอื่นได้ โดยมีหลักการคิดเหมือน ๆ กันเพียงแต่มอง ลักษณะการตัดให้

เข้าใจและเขียนองค์ประกอบสมการขึ้นมาจากภาพที่เห็น แล้วค่อยแทนค่าเหล่านั้นด้วยค่าตัวแปร ก็จะได้สมการในการหาตำแหน่งอาเรย์ 3 มิติได้ ส่วนสมการหาจำนวนช่อง และหาพื้นที่นั้น การตัดทั้ง 6 แบบจะมีสมการการหาเดียวกัน แต่สิ่งที่แตกต่าง กันมีเพียงสมการหาตำแหน่งอาเรย์ เท่านั้น ซึ่งการตัด 4 แบบที่เหลือ จะได้สมการดังต่อไปนี้

3.สมการการตัดแบบ “แถว-ระนาบ-สดมภ์” (Row-Plane-Column)

$$\begin{aligned}\text{Address} &= (\text{ตำแหน่งเริ่มต้น}) + (\text{จำนวนแถวที่ข้าม})(\text{จำนวนช่องต่อแถว})C + \\ &\quad (\text{จำนวนระนาบที่ข้าม})(\text{จำนวนช่องต่อระนาบ})C + (\text{จำนวนสดมภ์ที่ข้าม})C \\ &= BA + (j-l2)(u1-l1+1)(u3-l3+1)C + (i-l1)(u3-l3+1)C + (k-l3)C\end{aligned}$$

4.สมการการตัดแบบ “แถว-สดมภ์-ระนาบ” (Row-Column-Plane)

$$\begin{aligned}\text{Address} &= (\text{ตำแหน่งเริ่มต้น}) + (\text{จำนวนแถวที่ข้าม})(\text{จำนวนช่องต่อแถว})C + \\ &\quad (\text{จำนวนสดมภ์ที่ข้าม})(\text{จำนวนช่องต่อสดมภ์})C + (\text{จำนวนระนาบที่ข้าม})C \\ &= BA + (j-l2)(u1-l1+1)(u3-l3+1)C + (k-l3)(u1-l1+1)C + (i-l1)C\end{aligned}$$

5.สมการการตัดแบบ “สดมภ์-ระนาบ-แถว” (Column-Plane-Row)

$$\begin{aligned}\text{Address} &= (\text{ตำแหน่งเริ่มต้น}) + (\text{จำนวนสดมภ์ที่ข้าม})(\text{จำนวนช่องต่อสดมภ์})C + \\ &\quad (\text{จำนวนระนาบที่ข้าม})(\text{จำนวนช่องต่อระนาบ})C + (\text{จำนวนแถวที่ข้าม})C \\ &= BA + (k-l3)(u1-l1+1)(u2-l2+1)C + (i-l1)(u2-l2+1)C + (j-l2)C\end{aligned}$$

6.สมการการตัดแบบ “สดมภ์-แถว-ระนาบ” (Column-Row-Plane)

$$\begin{aligned}\text{Address} &= (\text{ตำแหน่งเริ่มต้น}) + (\text{จำนวนสดมภ์ที่ข้าม})(\text{จำนวนช่องต่อสดมภ์})C + \\ &\quad (\text{จำนวนแถวที่ข้าม})(\text{จำนวนช่องต่อแถว})C + (\text{จำนวนระนาบที่ข้าม})C \\ &= BA + (k-l3)(u1-l1+1)(u2-l2+1)C + (j-l2)(u1-l1+1)C + (i-l1)C\end{aligned}$$

1.3.3.2 โปรแกรมอาเรย์ 1-3 มิติ

```

/*
Program create array 1-3 dimension in function by..
1. Calculate and Allocate memory
2. Calculate the memoryaddress of array
3. Use point directed in to memory and read/write its
4. Formular useing
Element = (u-l+1)
Element = (u1-l1+1)*(u2-l2+1)
Element = (u1-l1+1)*(u2-l2+1)*(u3-l3+1)
Total_mem = Element*C
Address of Array
A(i) =BA+(i-l)C
A(i,j) =BA+(i-l1)*(u2-l2+1)C+(j-l2)C
A(i,j,k)=BA+(i-l1)*(u2-l2+1)(u3-l3+1)C+(j-l2)(u3-l3+1)C+(k-l3)C
=====*/
#include <stdio.h> //use printf()
#include <conio.h> //use getch()
#include <stdlib.h> //use malloc()
#define l 1 //lower Bound
#define u 5 //Upper Bound
#define l1 1 //lower Bound 1
#define u1 3 //Upper Bound 1
#define l2 1 //Lower Bound 2
#define u2 4 //Upper Bound 2
#define l3 1 //Lower Bound 3
#define u3 5 //Upper Bound 3
int *BA1, *BA2, *BA3, *p; //Base address of each dimension and moving
pointer
int i,j,k; //subscript of Array
void Create1DArray(){ //Create Array 1 dimension
    int element,c,total_mem; //Variable uses
    element=(u-l+1); //Calculate element
    c=sizeof(*BA1); //Calculate Size each block of Array
    total_mem=element*c; //Calculate Total Size
    BA1=(int*)malloc(total_mem); //Memory allocate and use BA1 point
its
}
void A1(int i,int x){ //Put data into Array 1 Dimension
    p=BA1+(i-l); //Calculate pointer
    *p=x; //Put data
}
int ReadA1(int i) { //Read data from Array 1 Dimension
    p=BA1+(i-l); //Calculate pointer
    return(*p); //Return value in Array
}

```

```

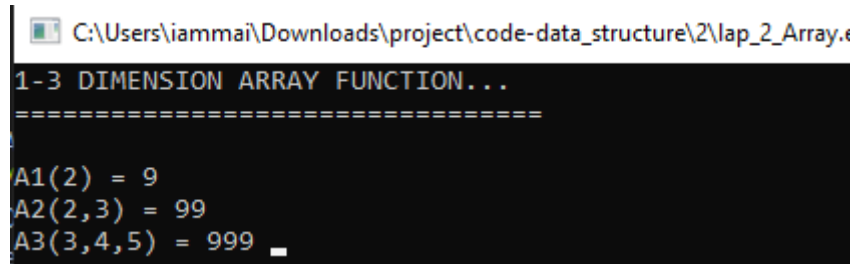
}
//-----
void Create2DArray() {
    int element,c,total_mem;
    element=(u1-l1+1)*(u2-l2+1);
    c=sizeof(*BA2);
    total_mem=element*c;
    BA2=(int*)malloc(total_mem);
}
void A2(int i,int j,int x) {
    p=BA2+((i-l1)*(u2-l2+1)+(j-l2));
    *p=x;
}
int ReadA2(int i,int j) {
    p=BA2+(i-l1)*(u2-l2+1)+(j-l2);
    return(*p);
}
//-----
void Create3DArray() {
    int element,c,total_mem;
    element=(u1-l1+1)*(u2-l2+1)*(u3-l3+1);
    c=sizeof(*BA3);
    total_mem=element*c;
    BA3=(int*)malloc(total_mem);
}
void A3(int i,int j,int k,int x) {
    p=BA3+((i-l1)*(u2-l2+1)*(u3-l3+1)+(j-l2)*(u3-l3+1)+(k-l3));
    *p=x;
}
int ReadA3(int i,int j,int k){
    p=BA3+(i-l1)*(u2-l2+1)*(u3-l3+1)+(j-l2)*(u3-l3+1)+(k-l3); ;
    return(*p);
}
//-----
int main() {
    printf("1-3 DIMENSION ARRAY FUNCTION...\n");
    printf("=====\n");
    // Create Array.....
    Create1DArray();
    Create2DArray();
    Create3DArray();
    //Using 1 Dimention Array...
    i=2;
    A1(i,9);
    printf("\nA1(%d) = %d ",i,ReadA1(i));
    //Using 2 Dimension Array ...

```

```

    i=2; j=3;
    A2(i,j,99);
    printf("\nA2(%d,%d) = %d ",i,j,ReadA2(i,j));
//Using 3 Dimension Array...
    i=3; j=4;k=5;
    A3(i,j,k,999);
    printf("\nA3(%d,%d,%d) = %d ",i,j,k,ReadA3(i,j,k));
    getch(); //Wait for KBD hit
    free(BA1); //Free memory of each array
    free(BA2);
    free(BA3);
    return(0);
} //End MAIN Fn.

```



```

C:\Users\iammai\Downloads\project\code-data_structure\2\lap_2_Array.c
1-3 DIMENSION ARRAY FUNCTION...
=====
A1(2) = 9
A2(2,3) = 99
A3(3,4,5) = 999

```

รูปที่ 1.21 แสดงผลการทำงานของโปรแกรมอาร์เรย์ 1-3 มิติ

แบบฝึกหัดที่ 1

1. ขนาดตัวชี้ตำแหน่ง (Addressing Pointer) ของหน่วยความจำมีขนาดกี่ไบต์แต่ละไบต์เก็บค่าอะไรบ้าง ?
2. การแทนข้อมูล (Data Representation) คืออะไร ?
3. ทำไมค่าลบของเลขจำนวนเต็ม (Integer) จึงมีค่ามากกว่าค่าบวกอยู่ 1 เสมอ จง อธิบายพร้อมแสดงตัวอย่างประกอบ ?
4. จงแสดงวิธีการหาค่าของ $129_{10} - 30_{10}$ โดยใช้วิธีการคิดเครื่องหมายโดยใช้ส่วนเติมเต็มสอง (Two's Complement Representation) มาตามลำดับ ?
5. จงแทนแสดงวิธีการแทนค่า -65.25_{10} ลงในรูปแบบมาตรฐาน IEEE 754 ?
6. การแทนข้อมูลชนิดตรรก ทำไมต้องใช้ถึง 8 บิต ทั้ง ๆ ที่ค่าทางตรรกมีเพียง 2 ค่า คือ “จริง” กับ “เท็จ” เท่านั้น จงอธิบายและให้เหตุผลประกอบ ?
7. จงแทนชื่อและนามสกุล(ภาษาอังกฤษ) ของแต่ละคนลงในหน่วยความจำ ?
8. อาร์เรย์ขนาด $A[5:10, 1:6, -9:2]$ แต่ละช่องเก็บตัวอักขระขนาด 4 ตัวอักษร ถูกจองไว้ในหน่วยความจำตั้งแต่ตำแหน่ง 0A0D:C2EB เป็นต้นไป จงคำนวณเพื่อตอบคำถาม ต่อไปนี้ ?
 - a) จำนวนไบต์ทั้งหมดที่ถูกจองในหน่วยความจำ
 - b) ตำแหน่งหน่วยความจำของอาร์เรย์ $A(8,5,-2)$ เมื่อเก็บแบบ Plane-Row-Column
 - c) ตำแหน่งหน่วยความจำของอาร์เรย์ $A(8,5,-2)$ เมื่อเก็บแบบ Row-Plane-Column
 - d) ตำแหน่งหน่วยความจำของอาร์เรย์ $A(8,5,-2)$ เมื่อเก็บแบบ Column-Row-Plane
9. จงแสดงวิธีการหาสมการในการหาตำแหน่งที่เก็บของอาร์เรย์ขนาด 3 มิติ โดยแสดงให้เห็นในแต่ละขั้นตอนอย่างละเอียดตามชนิดการเก็บดังต่อไปนี้ ?
 - a) Plane-Column-Row
 - b) Row-Plane-Column
 - c) Column-Plane-Row