CACHE

คุณลักษณะของหน่วยความจำในระบบคอมพิวเตอร์ (1)

Location Internal (e.g. processor registers, cache, main memory) External (e.g. optical disks, magnetic disks, tapes) **Capacity** Number of words Number of bytes **Unit of Transfer** Word Block **Access Method** Sequential Direct Random Associative	**Performance** Access time Cycle time Transfer rate **Physical Type** Semiconductor Magnetic Optical Magneto-optical **Physical Characteristics** Volatile/nonvolatile Erasable/nonerasable **Organization** Memory modules

คุณลักษณะของหน่วยความจำในระบบคอมพิวเตอร์ (2)

- Location
  - ใช้เรียกหน่วยความจำตามตำแหน่งที่อยู่ภายในหรือภายนอกเครื่องคอมพิวเตอร์
  - Internal memory ปกติจะหมายถึง หน่วยความจำหลัก
  - Processor มีหน่วยความจำของตัวเอง ก็คือ register
  - Cache ก็เป็นหน่วยความจำภายในอีกประเภทหนึ่ง
  - External memory ปกติจะหมายถึงอุปกรณ์เก็บข้อมูลที่ต้องเชื่อมต่อกับ processor ผ่านทาง I/O controller
- Capacity
  - ความจุของหน่วยความจำปกติของมีหน่วยเป็น byte
- Unit of transfer
  - สำหรับ internal memory หน่วยของการโอนย้ายข้อมูลจะเท่ากับจำนวนสายสัญญาณที่เข้า-ออก หน่วยความจำ

การเข้าถึงข้อมูลในหน่วยความจำ

- **Sequential access**
  - Memory มีการจัดรูปแบบของข้อมูลเรียกว่า records
  - การเข้าถึงข้อมูลจะเข้าถึงในรูปแบบของลำดับ
  - เวลาในการเข้าถึงข้อมูลไม่แน่นอน
  - หน่วยความจำประเภท Tape

- **Direct access**
  - เกี่ยวข้องกับกลไกการเขียน-อ่าน
  - แต่ละ block หรือ record จะมีหมายเลขแอดเดรสเฉพาะในการเข้าถึงข้อมูล
  - เวลาในการเข้าถึงข้อมูลไม่แน่นอน
  - ใช้ในหน่วยความจำประเภท Disk

- **Random access**
  - แต่ละตำแหน่งของหน่วยความจำจะมีแอดเดรสไม่ซ้ำกัน
  - เวลาในการเข้าถึงหน่วยความจำจะมีค่าแน่นอน
  - ไม่ว่าจะเป็นตำแหน่งไหนสามารถเข้าถึงข้อมูลได้โดยตรง
  - จะใช้สำหรับหน่วยความจำหลักและแคช

- **Associative**
  - ข้อมูลจะถูกดึงมาเป็นก้อนๆ มากกว่าอ้างถึงแอดเดรส
  - แต่ละตำแหน่งข้อมูลจะมีกลไลของแอดเดรสและรูปแบบการเข้าถึงที่แน่นอน
  - แคชส่วนใหญ่จะใช้วิธีการนี้ในการเข้าถึงข้อมูล

ประสิทธิภาพของหน่วยความจำ

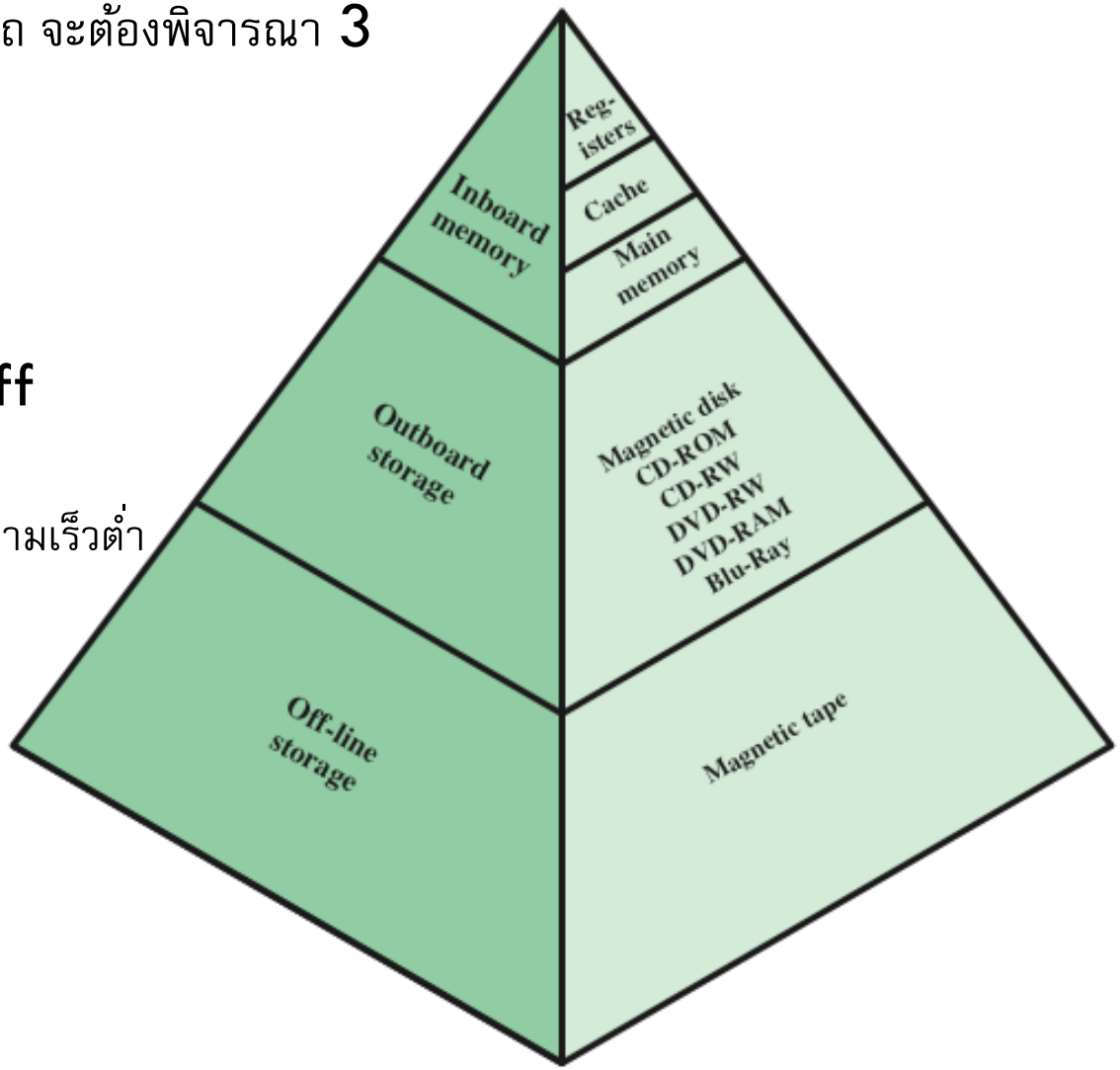
- ความจุ และ ประสิทธิภาพ
- ในการวัดประสิทธิภาพของหน่วยความจำจะมีข้อมูล 3 อย่างที่ต้องพิจารณา
  - Access time (latency)
    - สำหรับ random-access memory คือเวลาที่ใช้ในการอ่านหรือเขียน
    - สำหรับ ประเภทอื่นคือเวลาที่ใช้ในการเลื่อนหัวอ่านไปยังตำแหน่งที่ต้องการอ่านหรือเขียน
  - Memory cycle time
    - เวลาในการเข้าถึงข้อมูลบวกกับเวลาเพิ่มเติมที่ต้องใช้ก่อนที่การเข้าถึงข้อมูลถัดไปจะใช้ได้
    - เวลาเพิ่มเติมที่กล่าวถึงคือเวลาที่ให้สัญญาณที่ค้างอยู่ในสายสัญญาณหมดไปก่อน
    - เวลานี้จะเกี่ยวกับ System bus ไม่เกี่ยวข้องกับ Processor
  - Transfer rate
    - อัตราที่ข้อมูลสามารถโอนย้ายเข้า-ออกหน่วยความจำ
    - สำหรับ random-access memory จะมีค่าเท่ากับ 1/(cycle time)

หน่วยความจำ

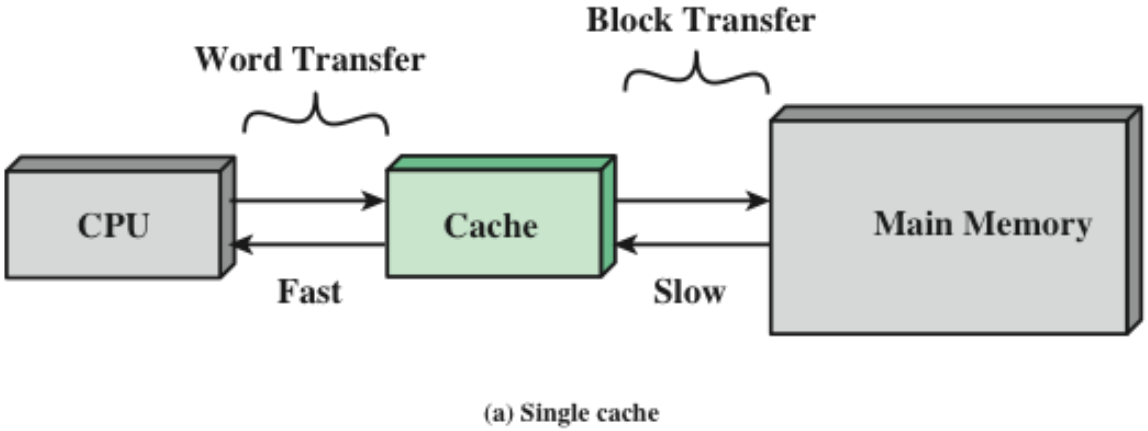
- รูปแบบของหน่วยความจำทั่วๆไป
  - หน่วยความจำที่ทำจาก Semiconductor
  - Magnetic surface
  - Optical
  - Magneto-optical
- คุณลักษณะทางกายภาพที่สำคัญในการเก็บข้อมูล
  - Volatile memory
    - ข้อมูลที่เก็บจะสูญหายถ้าไม่มีไฟฟ้ามาเลี้ยงหน่วยความจำ
  - Nonvolatile memory
    - เมื่อบันทึกข้อมูลแล้ว ข้อมูลจะไม่สูญหายแม้ไม่มีไฟฟ้ามาเลี้ยง แต่สามารถเปลี่ยนข้อมูลได้
  - Nonerasble memory
    - เป็นหน่วยความจำที่ไม่สามารถเปลี่ยนข้อมูลได้

ลำดับชั้นของหน่วยความจำ

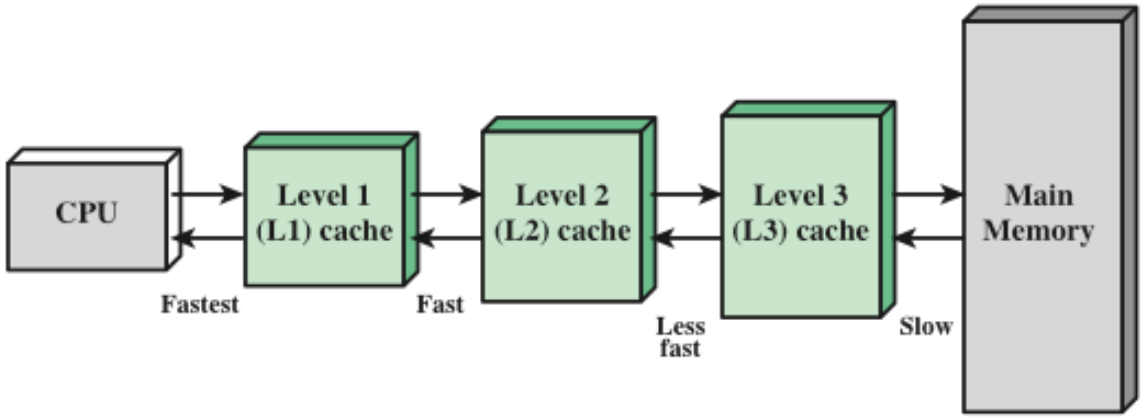
- การออกแบบหน่วยความจำสามารถ จะต้องพิจารณา 3 ส่วน คือ
  - ความจุ
  - ความเร็ว
  - ราคา
- ปกติการออกแบบจะมี trade-off
  - มีความเร็วสูง แต่มีราคาสูง
  - มีความจุสูง มีราคาต่ำ แต่มีการความเร็วต่ำ



Cache และหน่วยความจำหลัก



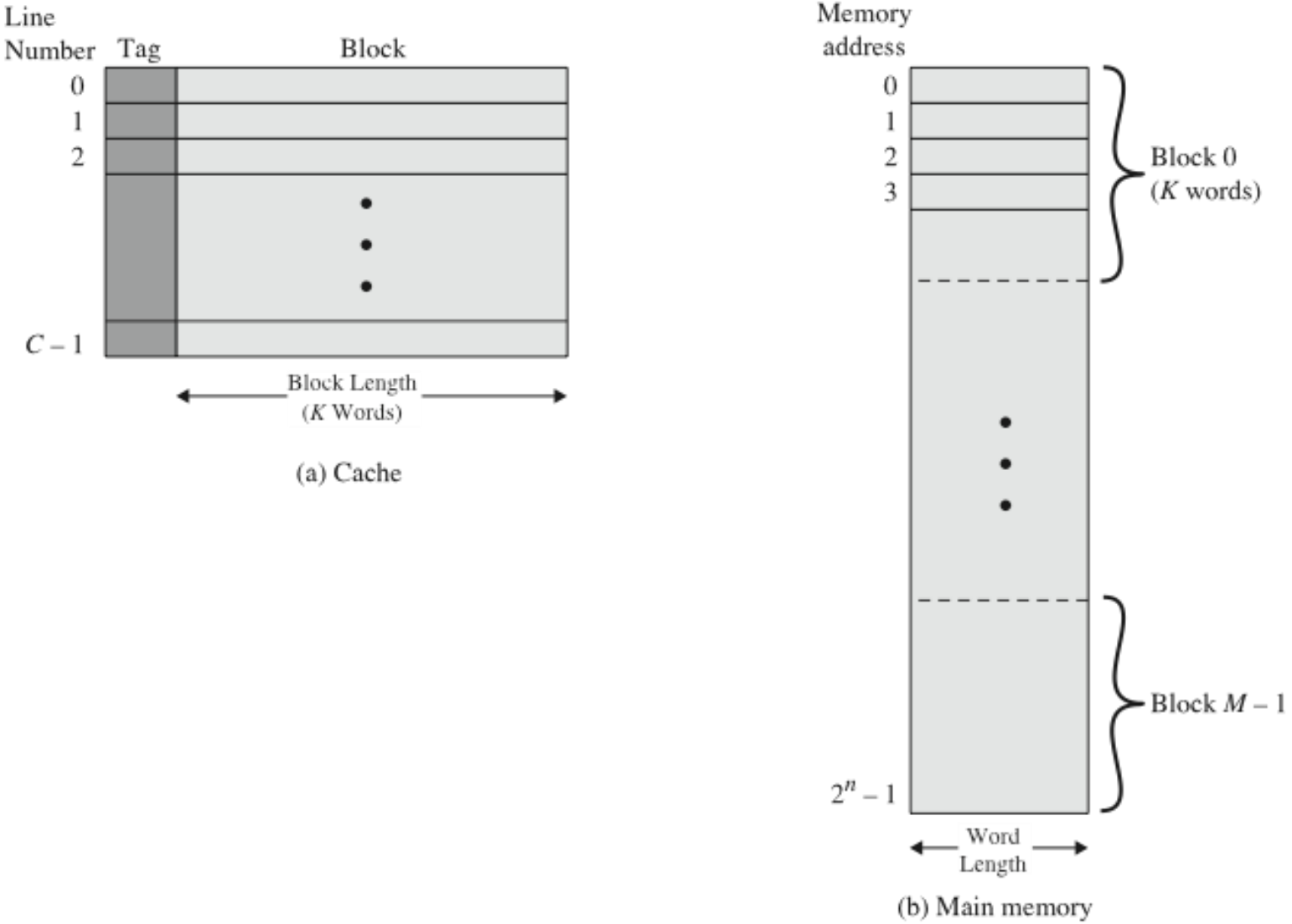
(a) Single cache



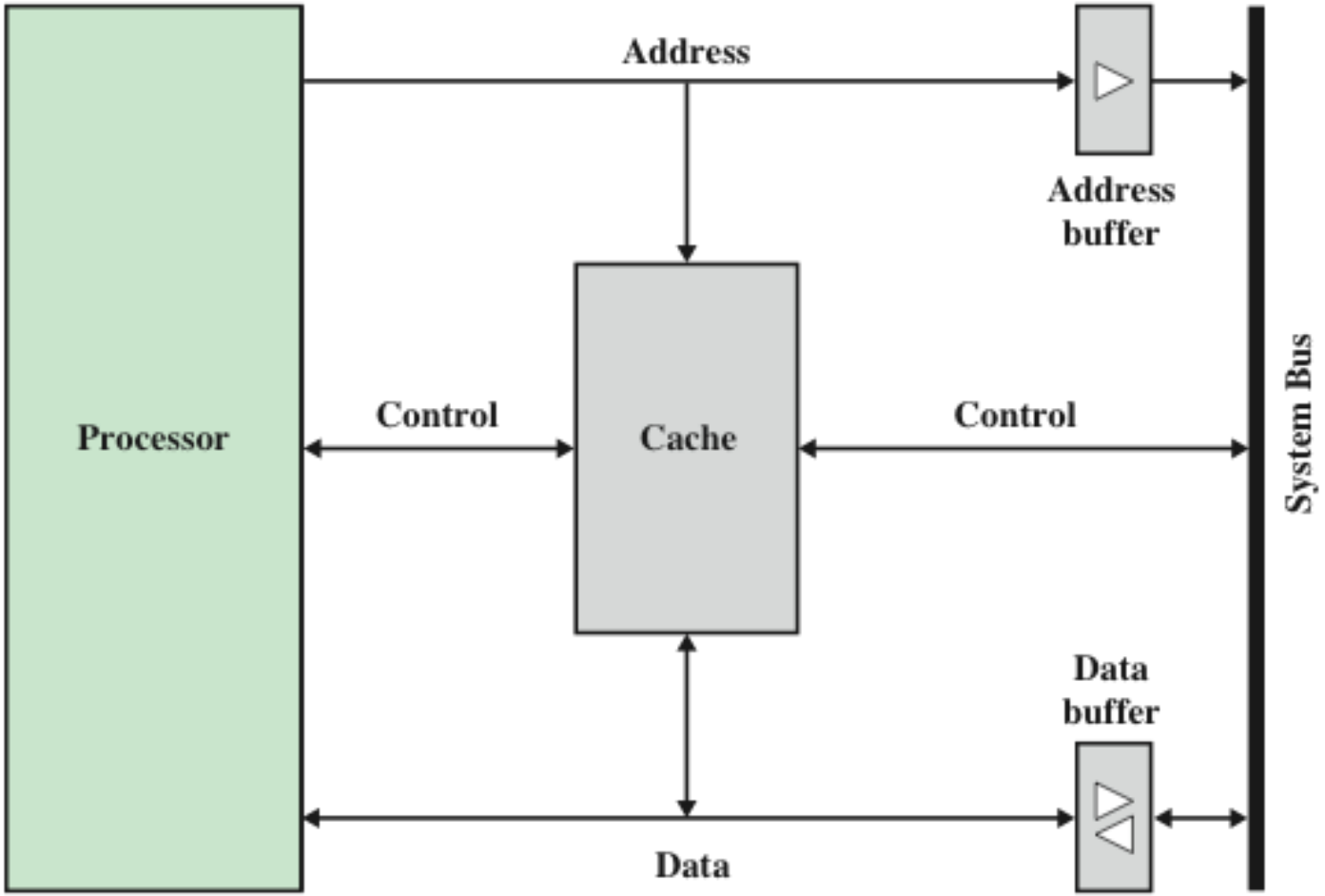
(b) Three-level cache organization

- Cache ถูกออกแบบมาให้มีความเร็วสูง และทำงานคู่กับหน่วยความจำหลักที่มีขนาดใหญ่กว่าแต่ราคาถูกกว่า

โครงสร้างของ Cache และหน่วยความจำหลัก

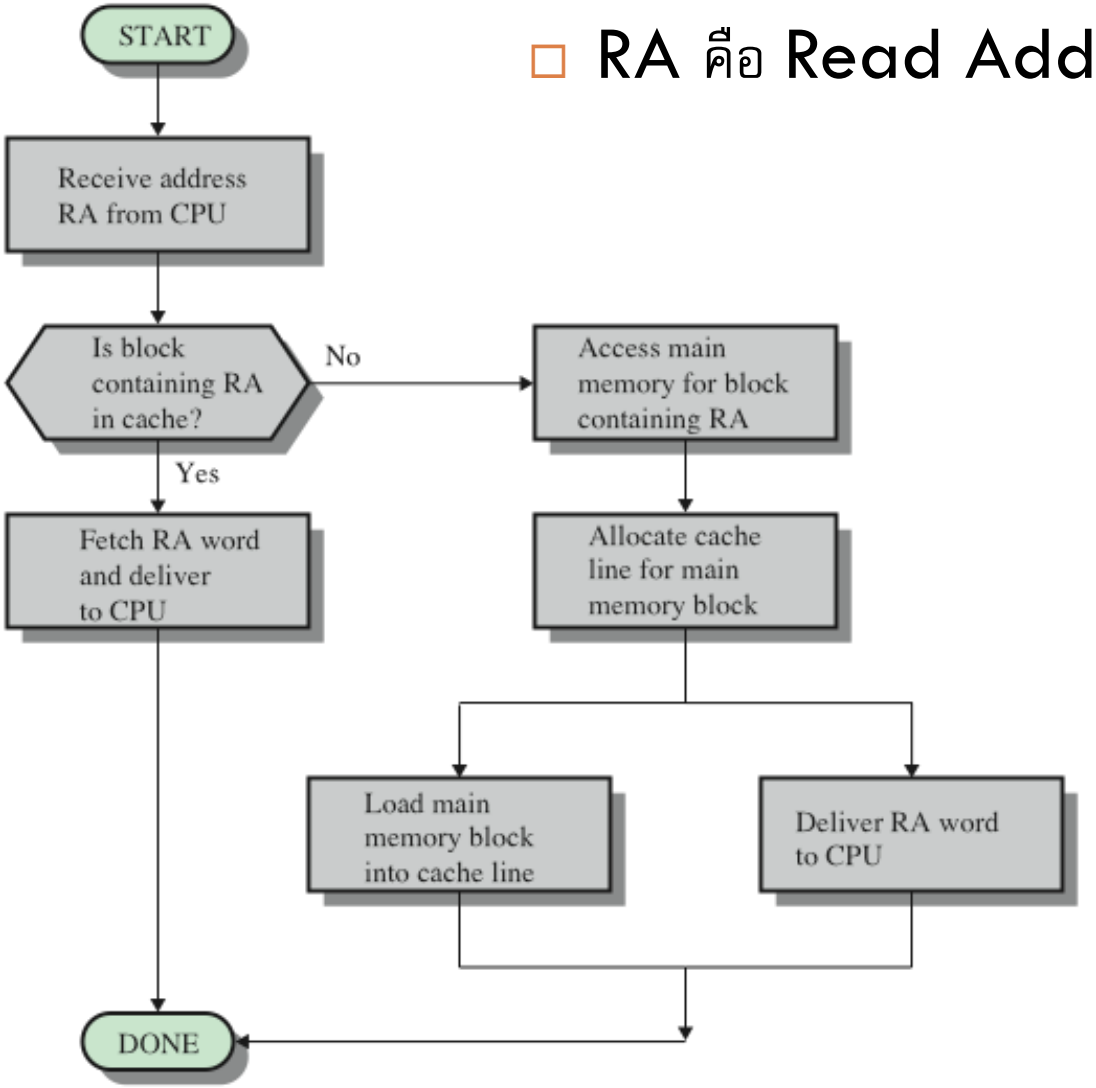


การวางตำแหน่ง Cache ในรูปแบบทั่วๆไป



Flowchart การอ่านข้อมูลจากเมื่อมี cache

- RA คือ Read Address



องค์ประกอบของ Cache

Cache Addresses
- Logical
- Physical

Cache Size

Mapping Function
- Direct
- Associative
- Set Associative

Replacement Algorithm
- Least recently used (LRU)
- First in first out (FIFO)
- Least frequently used (LFU)
- Random

Write Policy
- Write through
- Write back

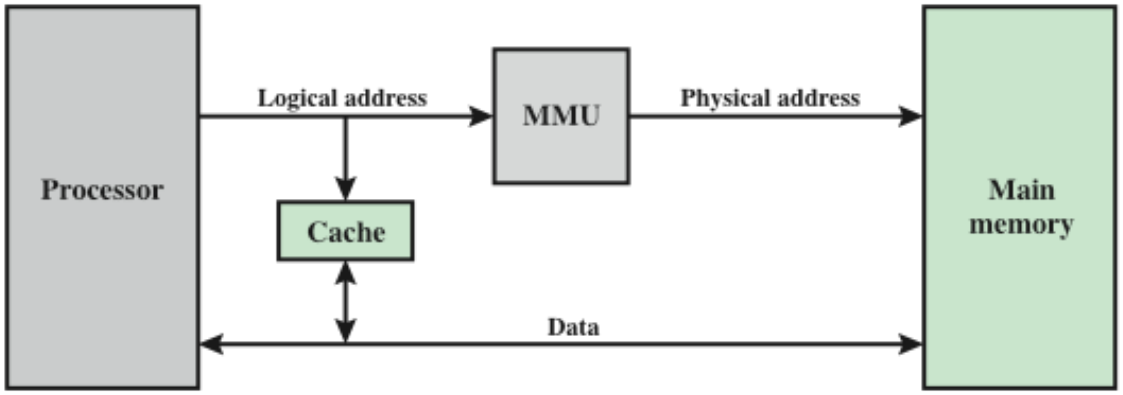
Line Size

Number of caches
- Single or two level
- Unified or split

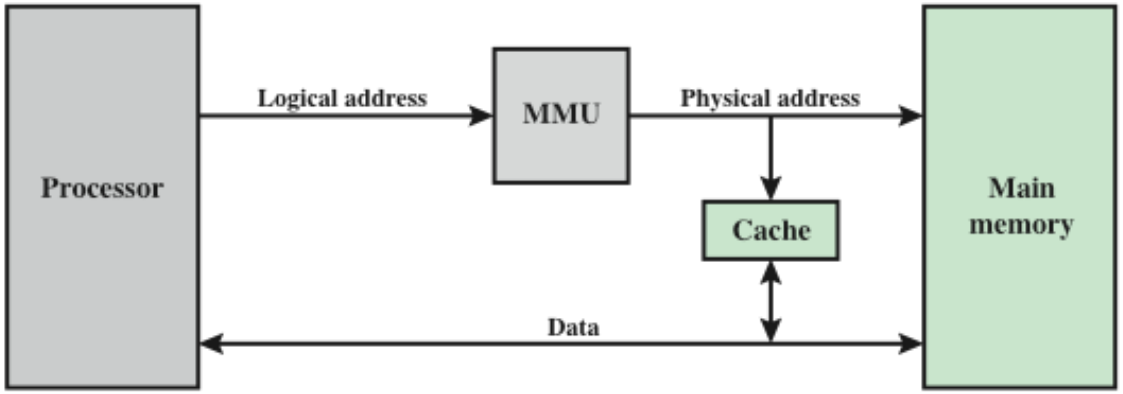
Cache Address

- Virtual Memory
  - เป็นตัวช่วยให้โปรแกรมสามารถอ้างอิงถึงหน่วยความจำหลักในมุมมอง **Logical** โดยไม่ต้องสนใจว่าหน่วยความจำหลักมีแอดเดรสอะไร
  - เมื่อมีการใช้งาน หมายเลขแอดเดรสในชุดคำสั่งจะกลายเป็น **virtual address**
  - สำหรับการเขียน/อ่าน ข้อมูลกับหน่วยความจำ จะมีฮาร์ดแวร์ชื่อ **memory management unit (MMU)** เป็นตัวแปลง **virtual address** ให้กลายเป็น **physical address** ของหน่วยความจำหลัก

ตำแหน่งการวาง Cache ที่นิยมใช้กัน



(a) Logical Cache



(b) Physical Cache

Cache Size

Processor	Type	Year of Introduction	L1 Cache[a]	L2 cache	L3 Cache
IBM 360/85	Mainframe	1968	16 to 32 kB	—	—
PDP-11/70	Minicomputer	1975	1 kB	—	—
VAX 11/780	Minicomputer	1978	16 kB	—	—
IBM 3033	Mainframe	1978	64 kB	—	—
IBM 3090	Mainframe	1985	128 to 256 kB	—	—
Intel 80486	PC	1989	8 kB	—	—
Pentium	PC	1993	8 kB/8 kB	256 to 512 KB	—
PowerPC 601	PC	1993	32 kB	—	—
PowerPC 620	PC	1996	32 kB/32 kB	—	—
PowerPC G4	PC/server	1999	32 kB/32 kB	256 KB to 1 MB	2 MB
IBM S/390 G6	Mainframe	1999	256 kB	8 MB	—
Pentium 4	PC/server	2000	8 kB/8 kB	256 KB	—
IBM SP	High-end server/ supercomputer	2000	64 kB/32 kB	8 MB	—
CRAY MTA[b]	Supercomputer	2000	8 kB	2 MB	—
Itanium	PC/server	2001	16 kB/16 kB	96 KB	4 MB
Itanium 2	PC/server	2002	32 kB	256 KB	6 MB
IBM POWER5	High-end server	2003	64 kB	1.9 MB	36 MB
CRAY XD-1	Supercomputer	2004	64 kB/64 kB	1MB	—
IBM POWER6	PC/server	2007	64 kB/64 kB	4 MB	32 MB
IBM z10	Mainframe	2008	64 kB/128 kB	3 MB	24-48 MB
Intel Core i7 EE 990	Workstaton/ server	2011	6 × 32 kB/32 kB	1.5 MB	12 MB
IBM zEnterprise 196	Mainframe/ Server	2011	24 × 64 kB/ 128 kB	24 × 1.5 MB	24 MB L3 192 MB L4

Mapping Function

- เพราะว่า cache มีขนาดเล็กกว่าหน่วยความจำหลัก ดังนั้นจึงต้องมีอัลกอริธึมในการ mapping block ของหน่วยความจำหลัก กับ Cache lines
- มีการใช้เทคนิคอยู่ 3 เทคนิค
  - Direct
    - เป็นเทคนิคที่ง่ายที่สุด
    - Map แต่ละ block ของหน่วยความจำหลักไปใน 1 cache line
  - Associative
    - อนุญาตให้แต่ละ block ของหน่วยความจำสามารถ load เข้าไปที่ line ไหนก็ได้ของ cache
    - Cache มีการอ่านสัญญาณควบคุมและแปลความหมายของแอดเดรสเป็น tag และ word
    - ในการตรวจสอบว่า block ของหน่วยความจำอยู่ใน cache หรือไม่ ตัวแปลความหมายจะตรวจสอบกับ Tag ของ cache ทุก line
  - Set Associative
    - เป็นเทคนิคที่รวมข้อดีของ Direct และ Associative

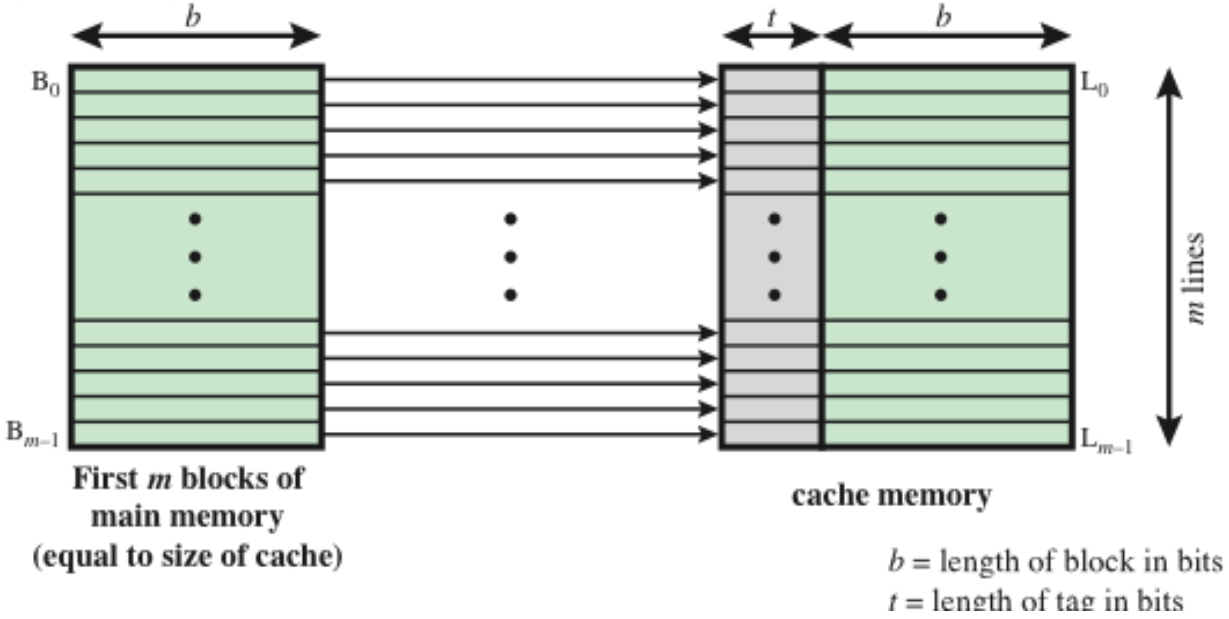
Direct Mapping

- เป็นเทคนิคที่ง่ายที่สุด โดยการ map แต่ละ block ของหน่วยความจำหลัก เข้าไปอยู่ใน 1 cache line จากสมการ

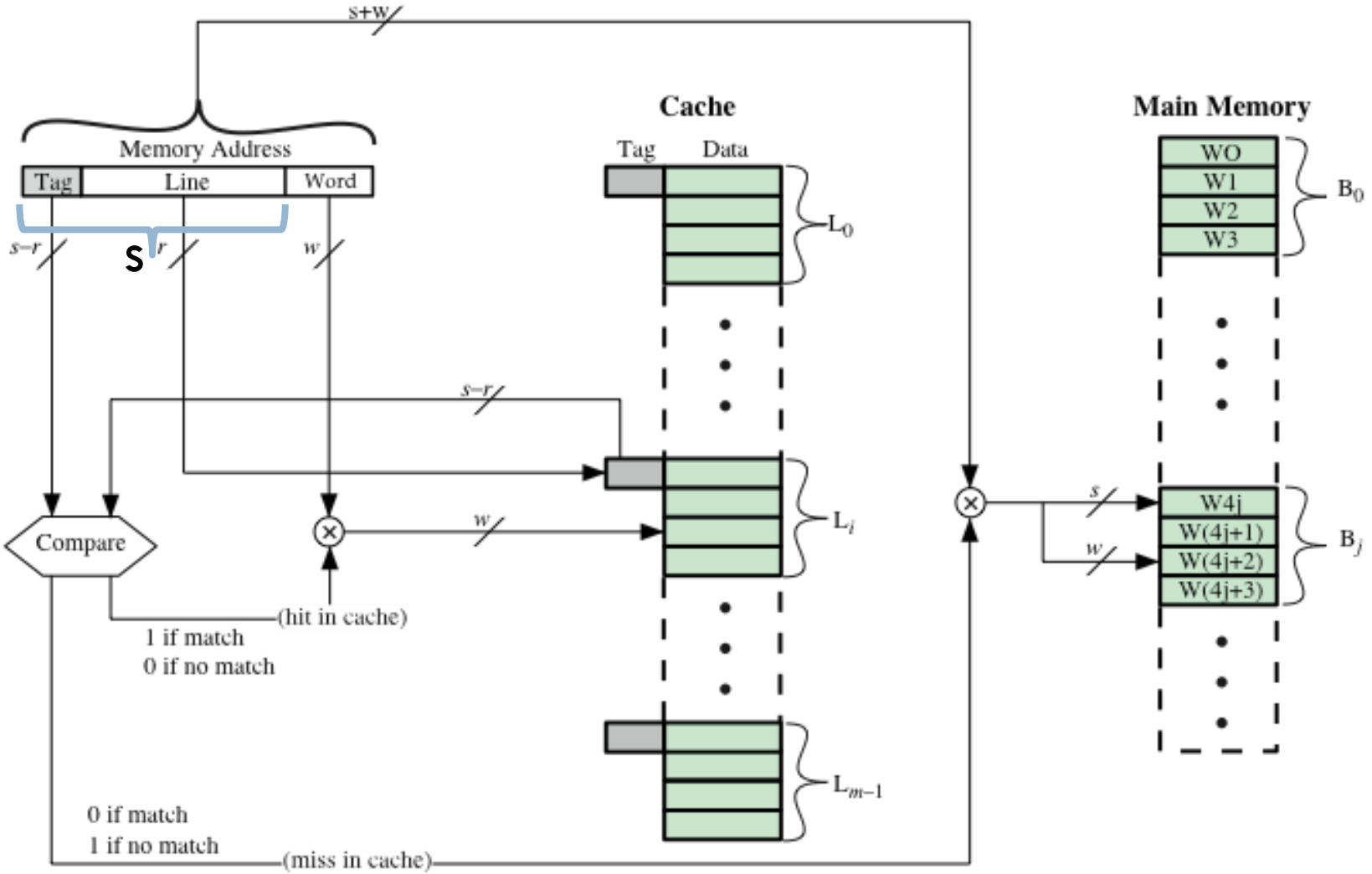
$$i = j \bmod m$$

ซึ่ง

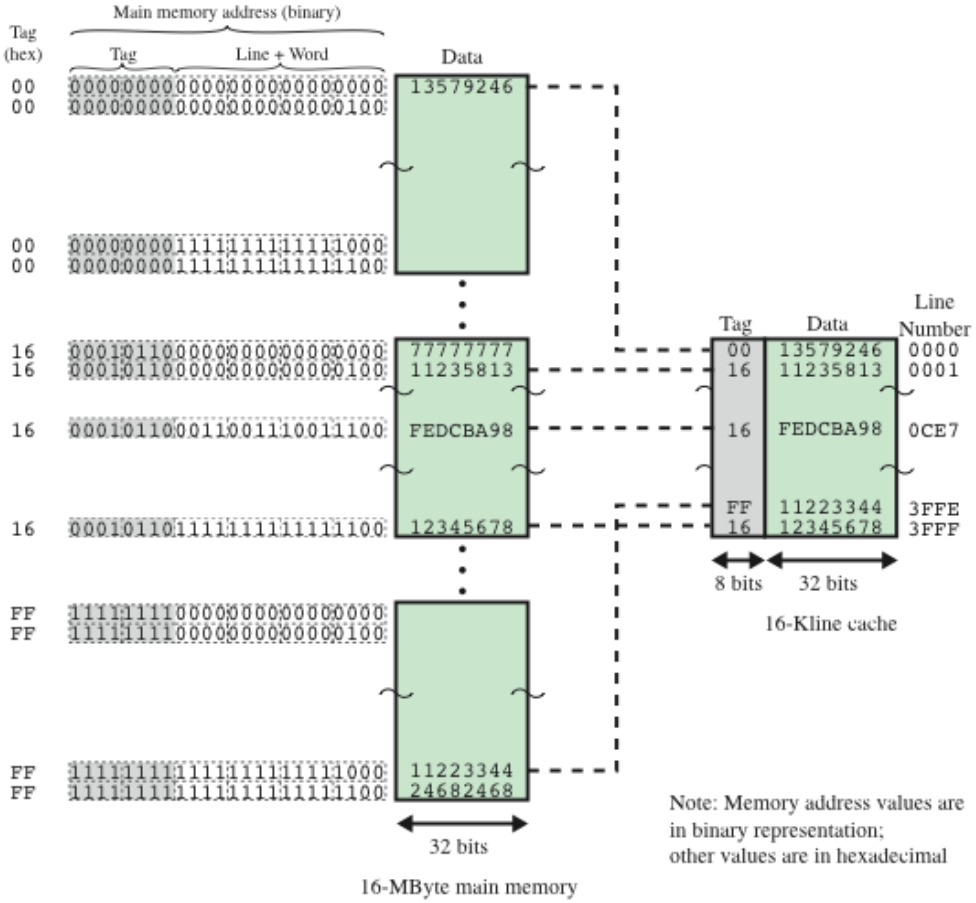
- i คือ หมายเลขของ cache line
- j คือ หมายเลข block ของหน่วยความจำหลัก
- m คือ จำนวนของ cache line

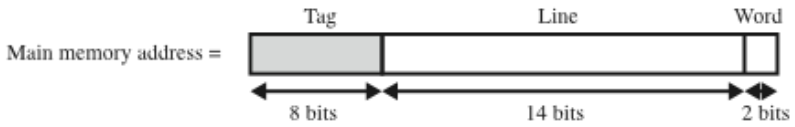


การทำงานภายในของเทคนิค Direct Cache



ตัวอย่างการใช้งาน Direct Cache





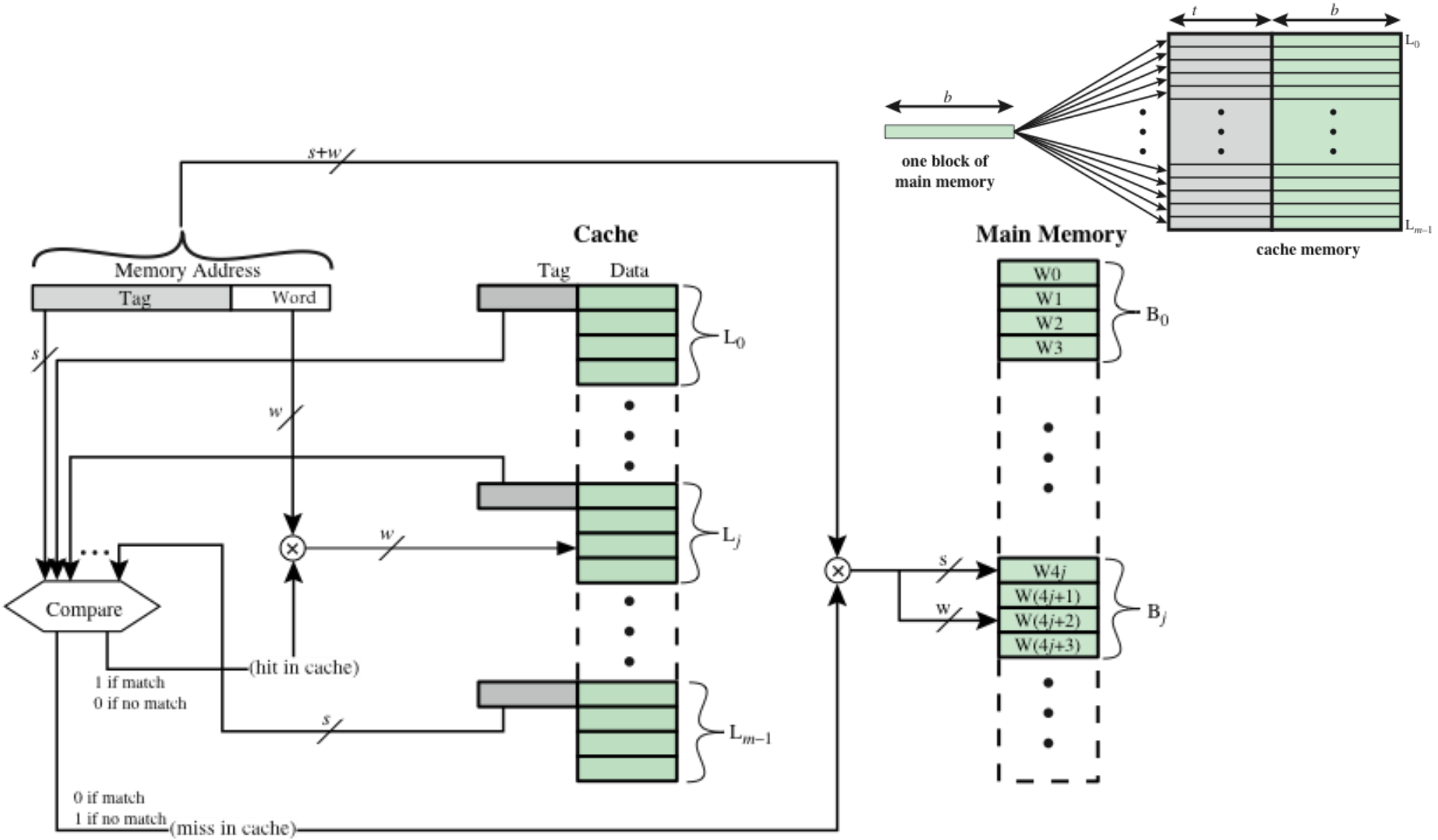
สรุป Direct Mapping

- ความยาว Address ของ cache = (s + w) bits
- จำนวนของแอดเดรสที่อ้างถึงได้ = 2^{s+w} words หรือ bytes
- Block size = line size = 2^{w} words หรือ bytes
- จำนวนของ block ในหน่วยความจำหลัก = 2^{s}
- จำนวนของ line ใน cache = 2^{r}
- ขนาดของ cache = 2^{r+w} words หรือ bytes
- ขนาดของ cache tag = (s – r) bits

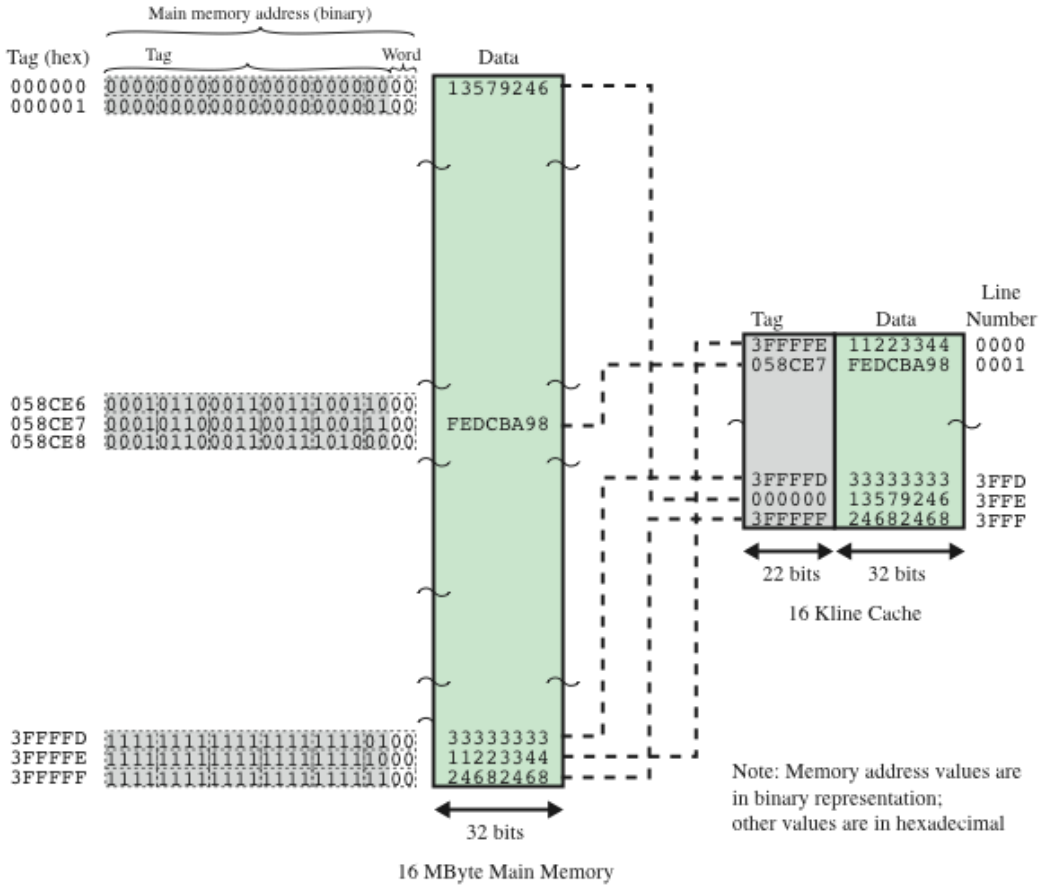
Victim Cache

- ถูกนำเสนอมาเพื่อแก้ไขปัญหาของ **direct mapped cache** ในกรณีที่ **line number** ซ้ำกัน
- มีการทำงานแบบ **Fully associative cache**
- ปกติจะมีขนาดเพียงแค่ **4 – 16 lines**
- จะวางในตำแหน่งระหว่าง **direct mapped cache** และ **memory** ชั้นถัดไป

Fully Associative Cache



ตัวอย่างของ Fully Associative Cache



สรุป Fully Associative Cache

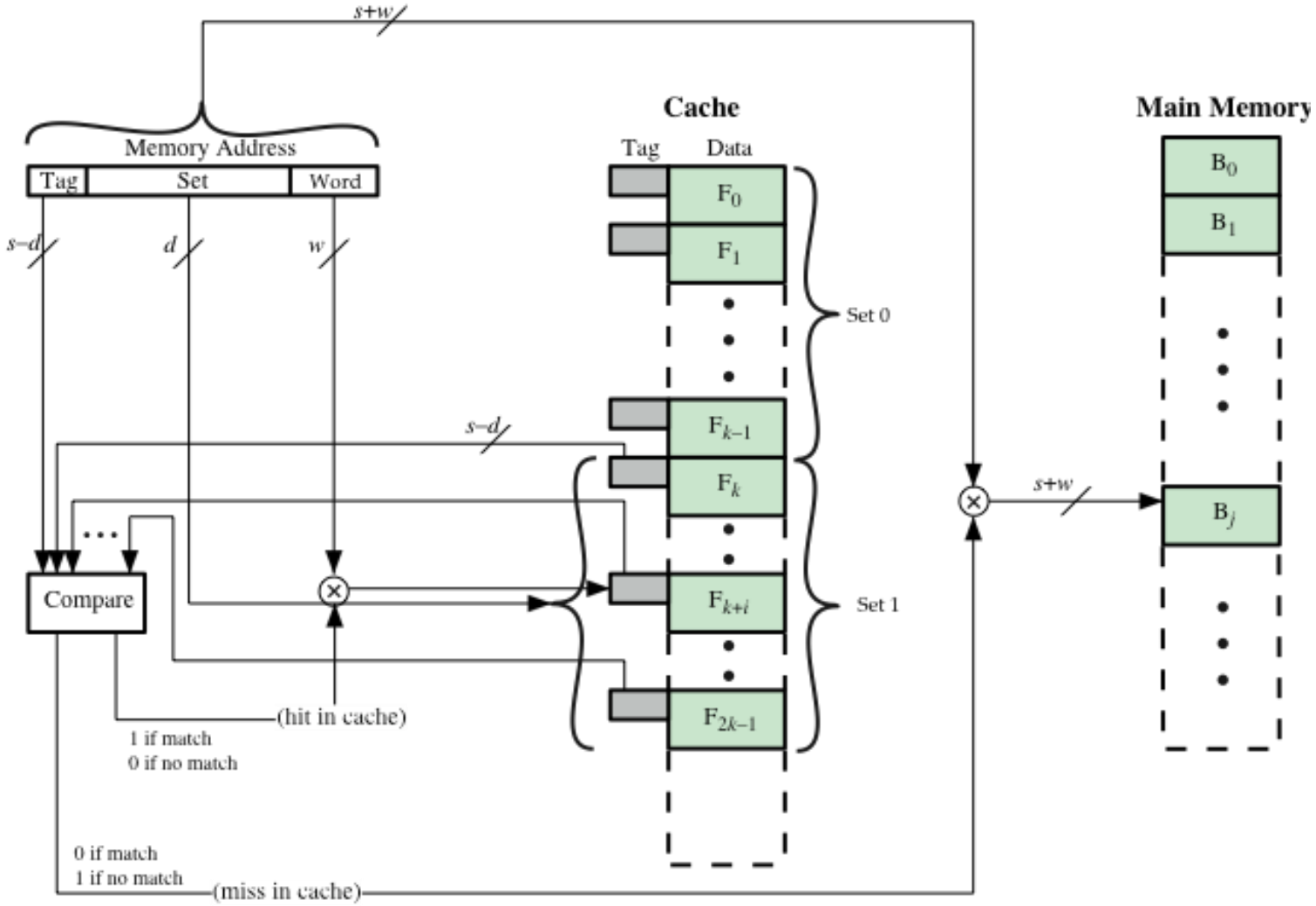
- ความยาว Address ของ cache = (s + w) bits
- จำนวนของแอดเดรสที่อ้างถึงได้ = 2^{s+w} words หรือ bytes
- Block size = line size = 2^{w} words หรือ bytes
- จำนวนของ block ในหน่วยความจำหลัก = 2^{s}
- จำนวนของ line ใน cache = unlimited
- ขนาดของ cache = unlimied
- ขนาดของ cache tag = s bits

Set Associative Mapping

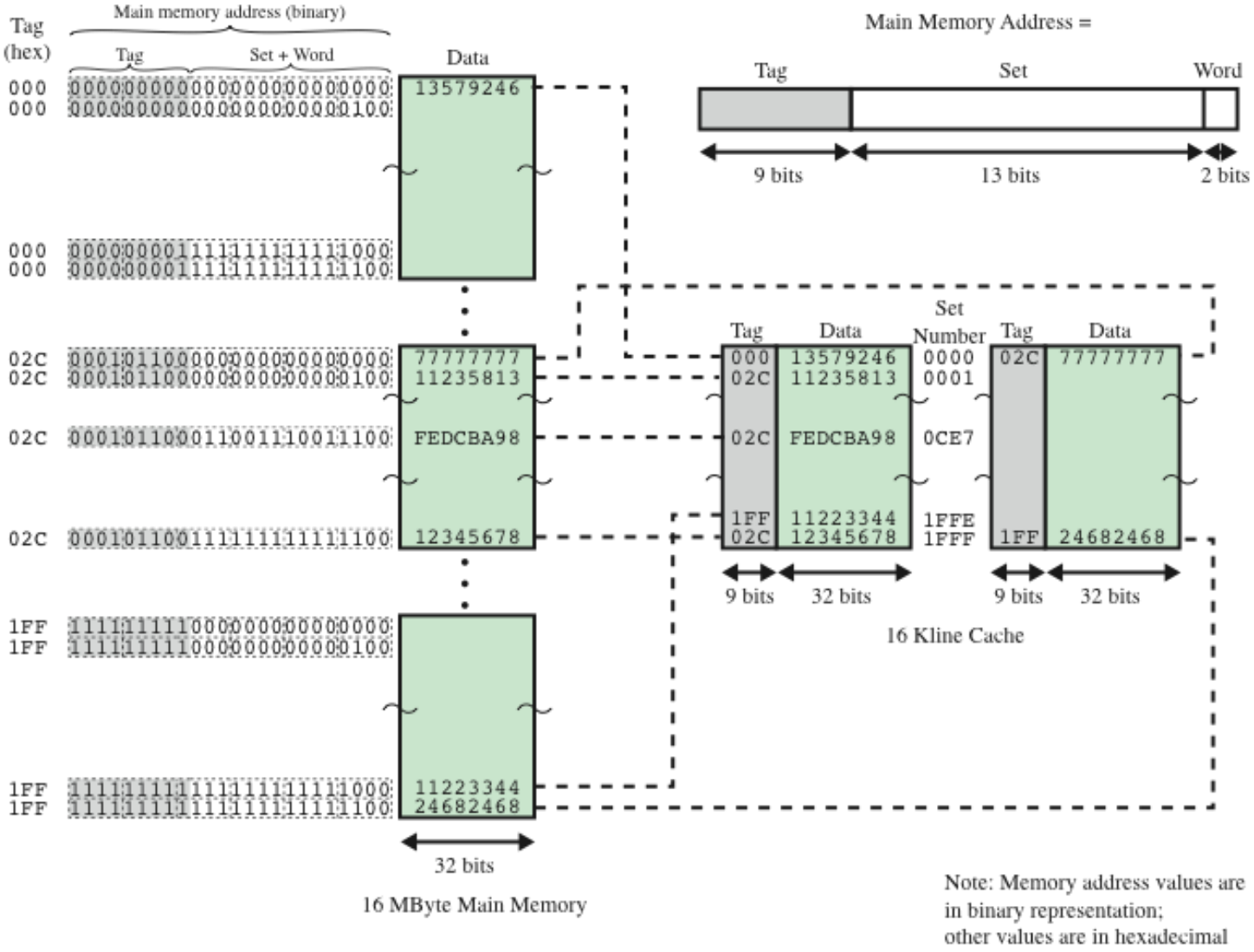
- เป็นการนำข้อดีของเทคนิค Direct map และ Fully Associative map มารวมกัน
- Cache จะประกอบไปด้วยจำนวนของ Set
- แต่ละ Set ประกอบไปด้วยจำนวนของ Lines
- Block ของหน่วยความจำจะถูก map ลงที่ line ไหนก็ได้ (fully associative) แต่อยู่ใน set ที่กำหนด (direct)
- กำหนดให้ $m = v * k$

  $i = j \bmod v$
  - i คือ หมายเลข set ของ cache
  - j คือ หมายเลข block ของหน่วยความจำหลัก
  - m คือ จำนวน line ของ cache
  - v คือ จำนวน set
  - k คือ จำนวน line ในแต่ละ set
- ส่วนใหญ่จะเรียกว่ากันว่า k-way set-associative mapping

การทำงานของ Set Associative Cache



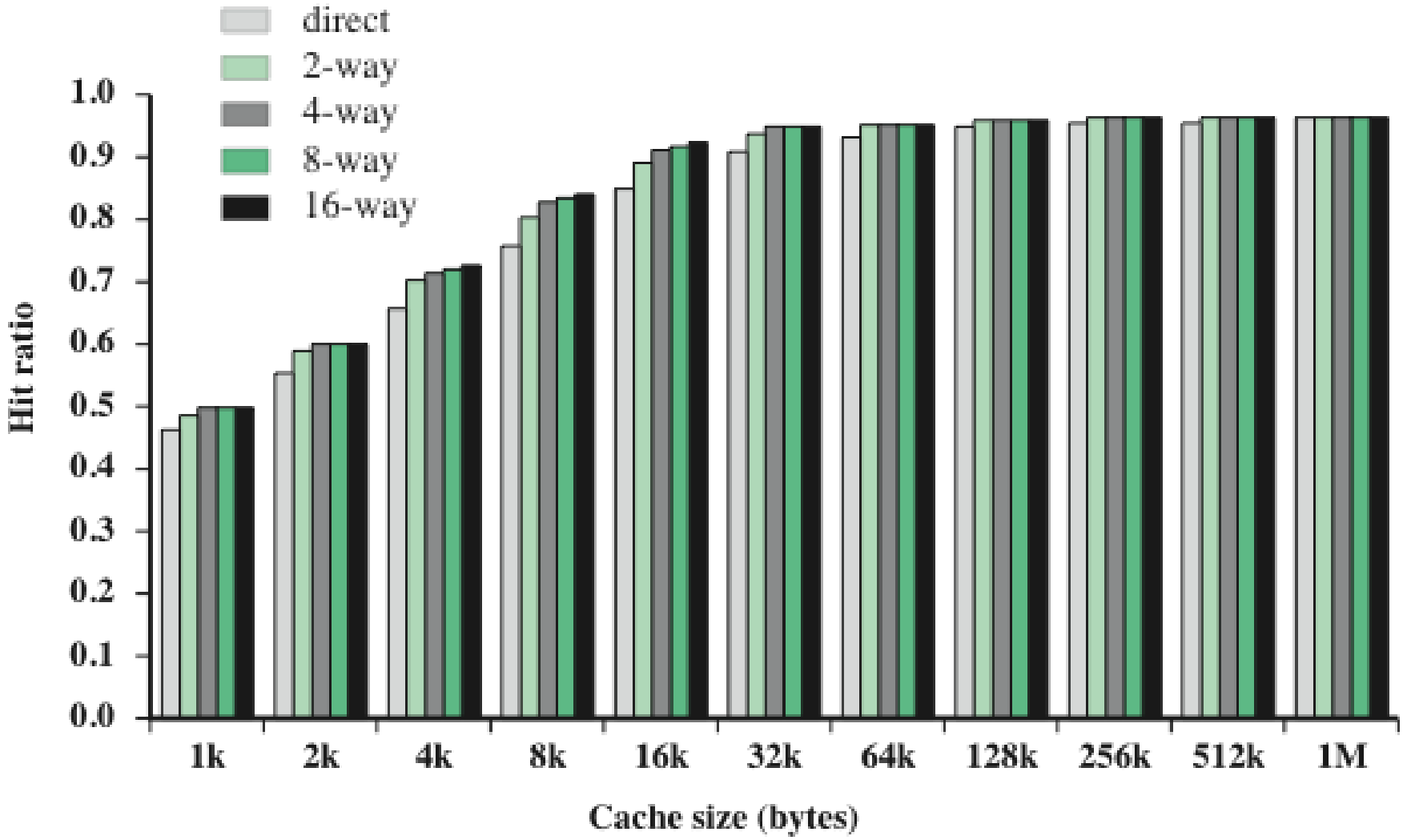
ตัวอย่าง การทำงานของ Set Associative Cache



สรุป Set Associative Cache

- ความยาว Address ของ cache = (s + w) bits
- จำนวนของแอดเดรสที่อ้างถึงได้ = 2^{s+w} words หรือ bytes
- Block size = line size = 2^{w} words หรือ bytes
- จำนวนของ block ในหน่วยความจำหลัก = 2^{s}
- จำนวนของ line ใน set = k
- จำนวนของ set = v = 2^{d}
- จำนวนของ line ใน cache = m = k * v = k * 2^{d}
- ขนาดของ cache = k * 2^{d+w} words หรือ bytes
- ขนาดของ cache tag = (s – d) bits

ประสิทธิภาพการทำงานของ Cache



Replacement Algorithm

- เมื่อ **cache** ได้เก็บข้อมูลเอาไว้แล้ว และมีข้อมูล **block** ใหม่ที่ต้องการเก็บใน **cache** จะทำให้ข้อมูลเก่าที่เก็บไว้จะต้องถูกแทนที่
- สำหรับ **direct mapping** เป็นการเก็บข้อมูลแบบตรง ตาม **line** ที่กำหนดดังนั้นจึงไม่มีทางเลือกอื่นนอกจากทับข้อมูลเก่าเลย
- สำหรับ **associative** และ **set-associative** นั้นมีควาจำเป็นที่จะต้องใช้อัลกอริธึมในการแทนที่ข้อมูล
- เพื่อความรวดเร็วในการทำงาน อัลกอริธึมนี้จะทำงานในรูปแบบของฮาร์ดแวร์

Replace Algorithm ที่นิยมใช้งาน

- Least recently used (LRU)
  - ถือว่าเป็นอัลกอริธึมที่ใช้งานกันอย่างแพร่หลายที่สุด
  - มีการประยุกต์ใช้งาน USE bit ถ้า block ไหนใน set ถูกใช้งานจะตั้งค่า USE bit เป็น 1 และ block ของ set เดียวกันที่เหลือจะตั้งค่า USE bit เป็น 0
  - เมื่อมี block ข้อมูลใหม่ จะนำไปแทนที่ข้อมูลของ block ใน set ที่มี USE bit เป็น 0
- First-in First-out (FIFO)
  - แทนที่ block ใน set ตัวที่อยู่ใน cache มานานที่สุด
  - สามารถทำได้ง่าย โดยใช้เทคนิค round-robin หรือ circular buffer
- Least frequently used (LFU)
  - แทนที่ block ใน set ตัวที่มีการอ้างอิงใช้งานน้อยที่สุด
  - สามารถทำได้ โดยใช้ counter ในแต่ละ line ของ cache
- Random
  - สุ่ม Block ที่จะถูกแทนที่เลย โดยไม่สนใจสิ่นอื่น

Write Policy

- เมื่อ Block ของหน่วยความจำอยู่ใน Cache และกำลังจะถูกแทนที่ มี 2 กรณีที่ต้องคำนึงถึง
  - ถ้าข้อมูลใน Block เก่าไม่ได้ถูกเปลี่ยนค่า Block ใหม่สามารถเข้ามาแทนที่ได้เลย
  - แต่ถ้ามีการเขียนอย่างน้อย 1 ครั้งใน Block เก่า จำเป็นจะต้อง update ค่านั้นกลับไปยังหน่วยความจำหลัก ก่อนที่จะนำ Block ใหม่เข้ามาแทนที่
- เพราะฉะนั้นมีปัญหาที่จะเกิดขึ้นตามมาคือ
  - ถ้ามีอุปกรณ์มากกว่า 1 อุปกรณ์ต้องการข้อมูลจากหน่วยความจำหลัก
  - ปัญหาที่ซับซ้อนมากกว่านั้นคือเมื่อ Processor หลายตัวที่ต่อบน bus เดียวกัน และมี cache เป็นของตัวเอง ถ้า Processor หนึ่งมีการแก้ไขใน cache ของตัวเอง อาจจะทำให้ Processor อีกตัวนึงมีข้อมูลใน cache ที่ไม่ตรงกัน

Write Through และ Write Back

- Write through
  - เป็นเทคนิคที่ง่ายที่สุด
  - ทุกๆ การดำเนินการที่เป็นการ write จะเขียนทั้งใน cache และ หน่วยความจำหลัก
  - ข้อเสียหลักของเทคนิคนี้คือ จะทำให้เกิดการติดต่อกับหน่วยความจำหลักค่อนข้างเยอะ และทำให้เกิดปัญหาคอขวด
- Write back
  - พยายามลดปัญหาการติดต่อกับหน่วยความจำหลัก
  - มีการประยุกต์ใช้ Dirty bit ในแต่ละ line ของ cache
  - ถ้ามีการแทนที่ block จะตรวจสอบ dirty bit ถ้ามีถูก set จะเขียนข้อมูลกลับลงหน่วยความจำหลัก ก่อนที่จะแทนที่ block เก่าด้วย block ใหม่

การจัดการ Cache สำหรับ Processor หลายตัว

- ใน bus ถ้ามี processor มากกว่า 1 ตัว แต่ละตัวมี cache ของตัวเอง แต่หน่วยความจำหลักจะมีการใช้งานร่วมกัน จะมีปัญหาใหม่เกิดขึ้นมา คือ ถ้ามี processor ใด processor เปลี่ยนค่าใน cache ไม่เพียงแต่ข้อมูลในหน่วยความจำหลักที่ต้องจัดการ แต่ต้องจัดการกับข้อมูลใน cache ของอีก processor ด้วย
- Bus watching with write through
  - ตัวควบคุม cache จะคอยตรวจสอบแอดเดรสของ line นั้นว่ามีการเขียนข้อมูลไหม ถ้ามีการเขียนข้อมูลเกิดขึ้น จะส่งสัญญาณไปทำให้ cache ใน processor อื่น invalid
- Hardware transparency
  - เพื่อ hardware ที่มีกลไกการเปลี่ยนข้อมูลในหน่วยความจำและ cache ที่เกี่ยวข้องทั้งหมด
- Non-cacheable memory
  - อนุญาตให้แค่บางส่วนในหน่วยความจำหลักเท่านั้น ที่แบ่งปันระหว่าง Processor และส่วนนั้นจะถูกออกแบบมาไม่ให้มีการทำ cache

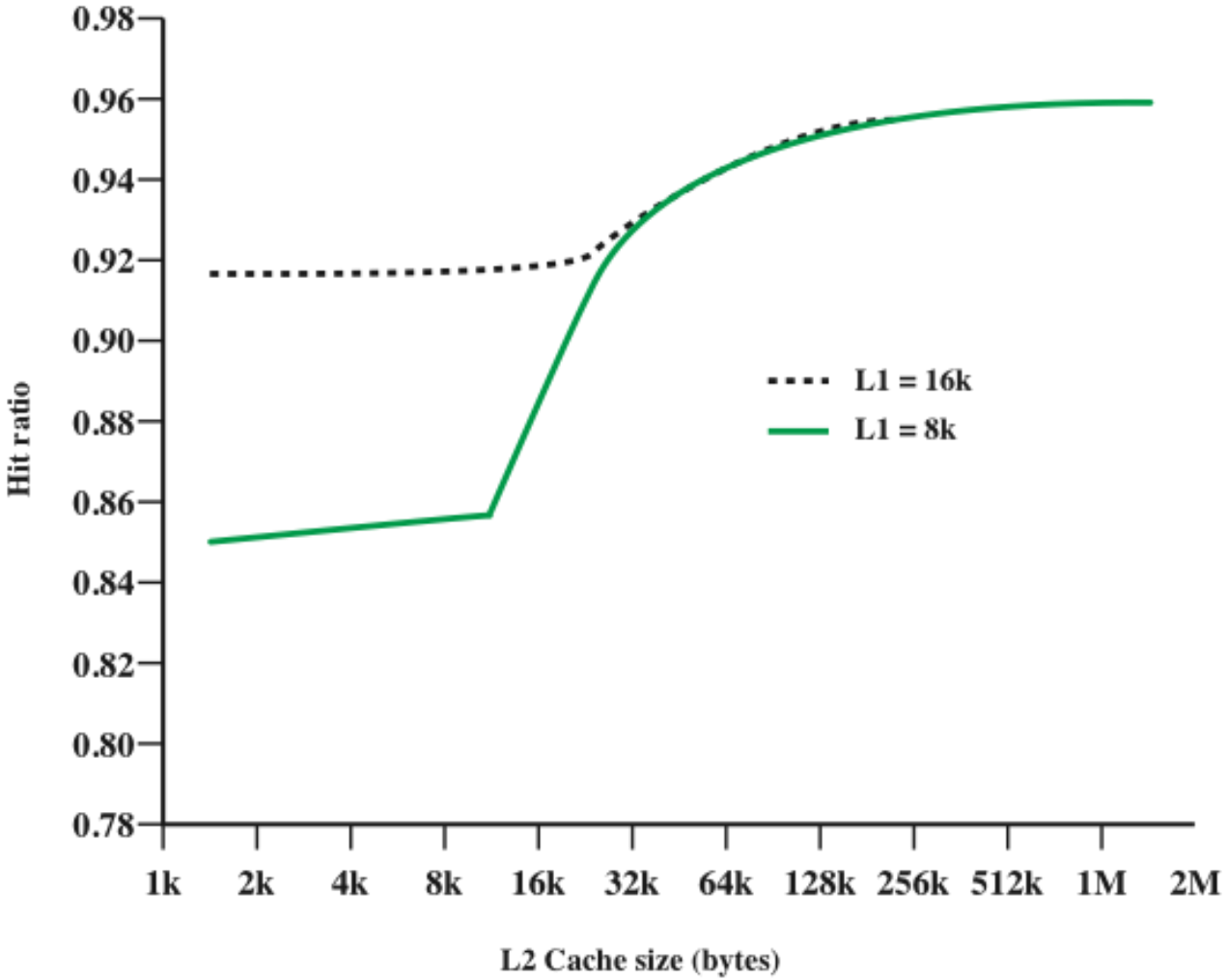
Line Size

- เมื่อ Block ข้อมูลเข้ามาเก็บใน Cache จะเก็บเป็นจำนวนเท่ากับขนาดของ line ทำให้มีข้อมูลหรือชุดคำสั่งที่ติดกัน ถูกดึงเข้ามาด้วย
- เมื่อขนาดของ line ใหญ่ขึ้น จะเพิ่มอัตราส่วนของ cache hit มากขึ้น จากหลักการของ locality
- แต่อย่างไรก็ตามอัตราส่วนของ cache hit จะลดลงถ้า line ใหญ่เกินไป และความน่าจะเป็นที่ดึงข้อมูลที่ไม่ได้ใช้งานเข้ามามีเยอะมากขึ้น
- ดังนั้นจึงมีคุณสมบัติที่มีผลกระทบอยู่ 2 อย่างสร้าง
  - ถ้า line มีขนาดใหญ่มากจะทำให้จำนวน line ลดลง
  - ถ้า line ใหญ่มากไปจะทำให้ข้อมูลที่ต้องใช้งานอยู่ไกลเกินไป
- จากงานวิจัยพบว่าขนาดของ line ที่ 8 – 64 bytes เหมาะสมกับการทำงานทั่วไป และสำหรับ HPC จะใช้ line ขนาด 64 – 128 bytes

Multilevel Caches

- ความหนาแน่นของ gate ที่เพิ่มขึ้นใน chip สามารถทำให้สามารถทำ cache ไว้ภายใน chip ของ processor ได้
- การที่ cache อยู่ภายใน chip จะทำให้ลดการใช้งาน bus ภายนอกและเพิ่มความเร็วให้การทำงานของ processor
  - ถ้าพบชุดคำสั่งใน cache จะทำให้ไม่จำเป็นต้องใช้งาน bus เพื่อดึงชุดคำสั่งจากหน่วยความจำหลัก
  - การดึงข้อมูลจาก cache ภายใน processor จะเร็วมากแทบจะไม่ต้องรอสัญญาณ clock เลย
  - เมื่อ bus ไม่ได้ถูกใช้งานจะทำให้ bus สามารถให้บริการกับอุปกรณ์อื่นแทนได้
- Two-level cache :
  - Internal cache ออกแบบมาอาจจะเรียกว่า level 1 (L1)
  - External cache ออกแบบมาทำงานถัดจาก level 1 เรียกว่า level 2 (L2)

Hit Ratio (8KB และ 16KB L1)



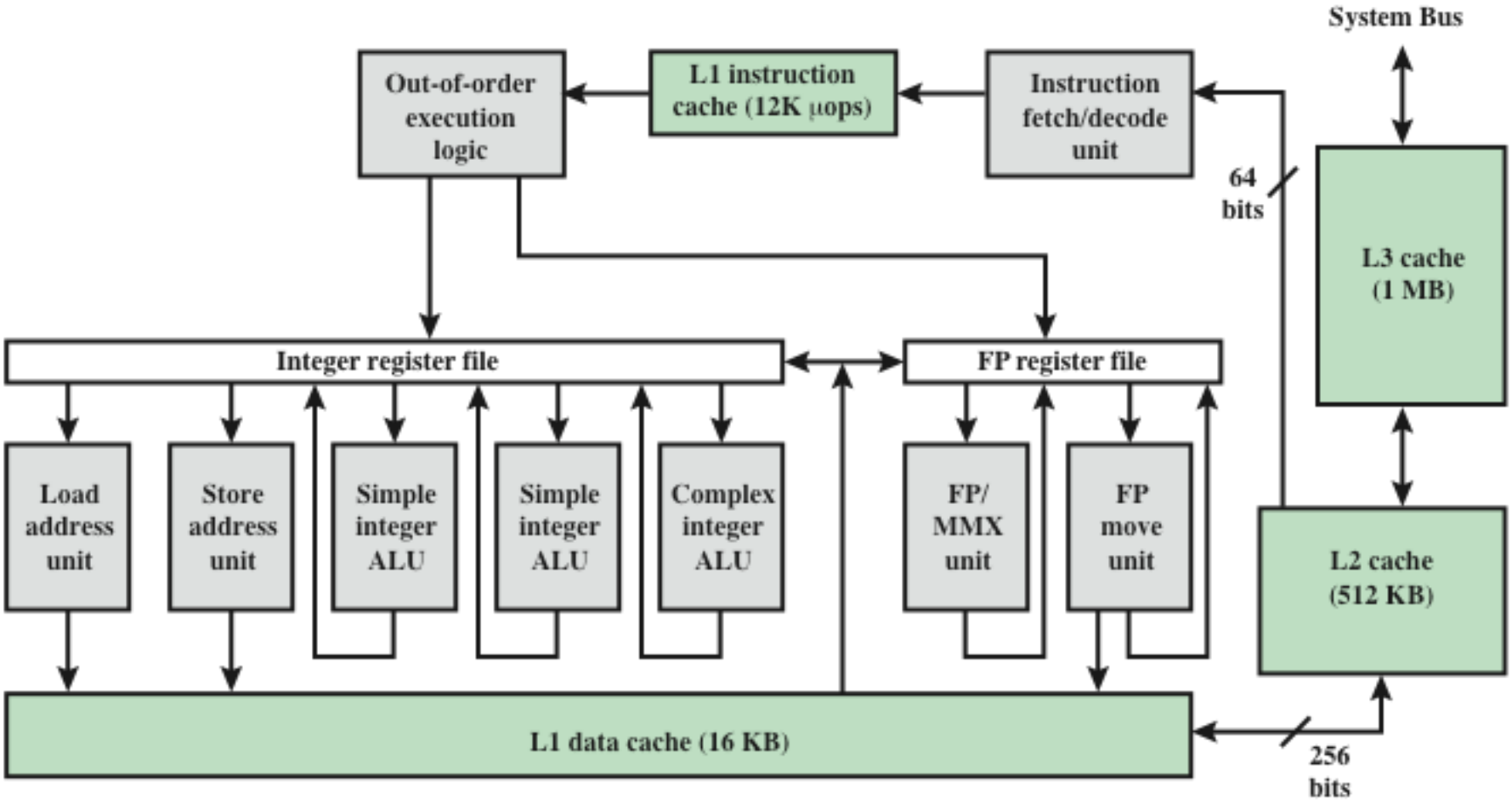
Unified Cache กับ Split Cache

- ข้อดีของ Unified cache
  - มีอัตราการ hit ที่สูง
    - เนื่องจากเป็นการรวมกันระหว่างชุดคำสั่งและชุดข้อมูลที่ต้องการจะใช้งาน
    - การออกแบบและพัฒนาทำกับ cache เพียงแค่ตัวเดียว
- Split cache จะมี
  - Instruction cache สำหรับเก็บชุดคำสั่ง
  - Data cache สำหรับเก็บชุดข้อมูล
  - โดยทั้ง 2 cache นี้จะอยู่ในระดับเดียวกัน ปกติจะเป็น L1 ทั้งคู่
- ข้อดีของ Split cache
  - ลดความหนาแน่นของการใช้งาน cache ระหว่าง fetch, decode และ execute
    - ซึ่งจะสำคัญมากกับการทาง Pipelining
- แนวโน้มการออกแบบจะใช้ Split cache สำหรับ L1 และ Unified cache สำหรับ level ที่สูงกว่า

Pentium Cache History

Problem	Solution	Processor on which Feature First Appears
External memory slower than the system bus.	Add external cache using faster memory technology.	386
Increased processor speed results in external bus becoming a bottleneck for cache access.	Move external cache on-chip, operating at the same speed as the processor.	486
Internal cache is rather small, due to limited space on chip	Add external L2 cache using faster technology than main memory	486
Contention occurs when both the Instruction Prefetcher and the Execution Unit simultaneously require access to the cache. In that case, the Prefetcher is stalled while the Execution Unit's data access takes place.	Create separate data and instruction caches.	Pentium
Increased processor speed results in external bus becoming a bottleneck for L2 cache access.	Create separate back-side bus that runs at higher speed than the main (front-side) external bus. The BSB is dedicated to the L2 cache.	Pentium Pro
	Move L2 cache on to the processor chip.	Pentium II
Some applications deal with massive databases and must have rapid access to large amounts of data. The on-chip caches are too small.	Add external L3 cache.	Pentium III
	Move L3 cache on-chip.	Pentium 4

Pentium 4 Block Diagram



ARM Cache

Core	Cache Type	Cache Size (kB)	Cache Line Size (words)	Associativity	Location	Write Buffer Size (words)
ARM720T	Unified	8	4	4-way	Logical	8
ARM920T	Split	16/16 D/I	8	64-way	Logical	16
ARM926EJ-S	Split	4-128/4-128 D/I	8	4-way	Logical	16
ARM1022E	Split	16/16 D/I	8	64-way	Logical	16
ARM1026EJ-S	Split	4-128/4-128 D/I	8	4-way	Logical	8
Intel StrongARM	Split	16/16 D/I	4	32-way	Logical	32
Intel Xscale	Split	32/32 D/I	8	32-way	Logical	32
ARM1136-JF-S	Split	4-64/4-64 D/I	8	4-way	Physical	32