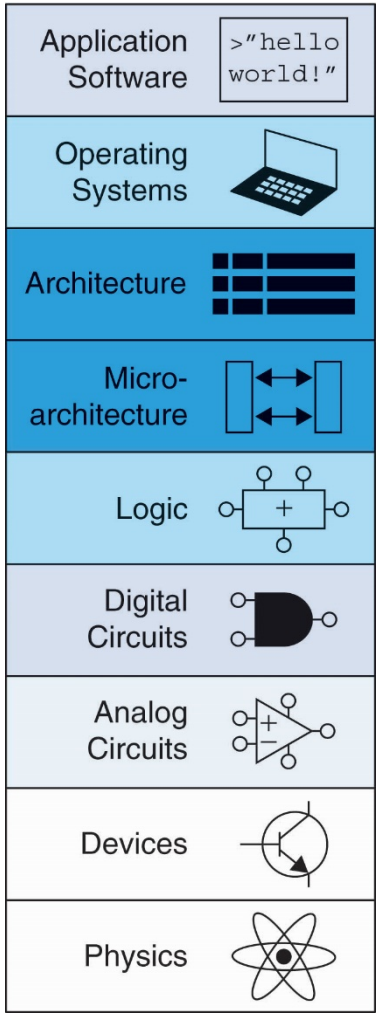
Розділ 8. Теми

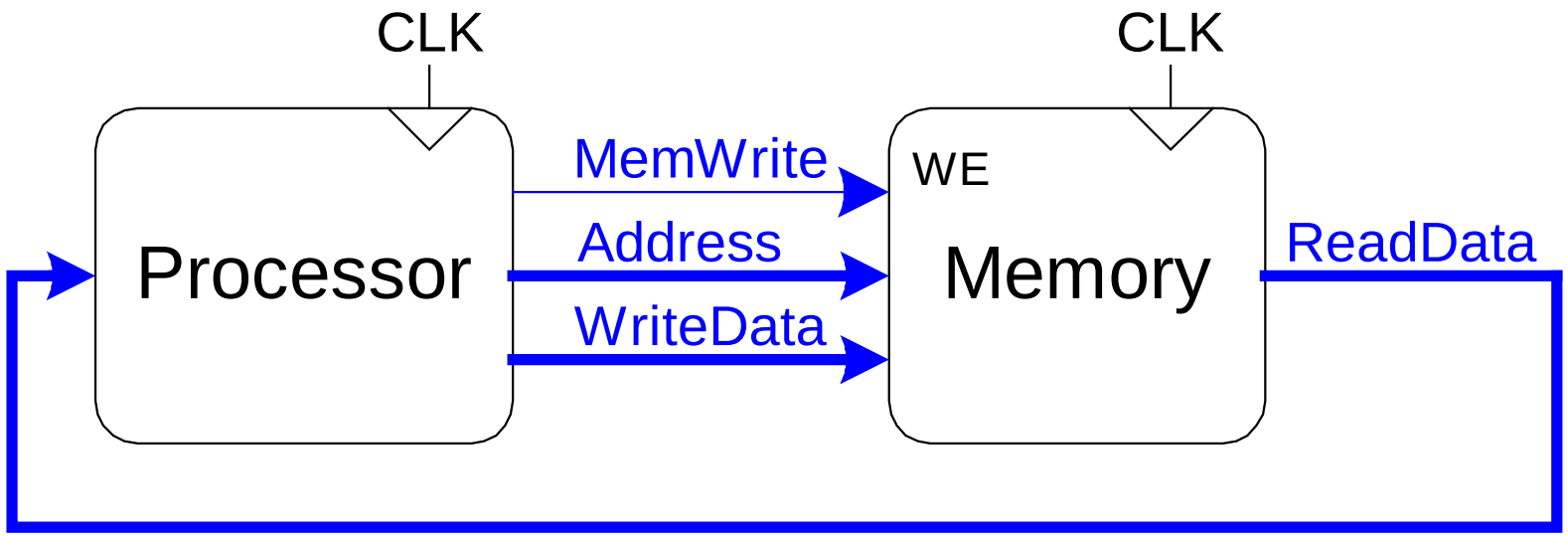
- Вступ
- Аналіз продуктивності систем пам’яті
- Кеш-пам’ять
- Віртуальна пам’ять
- Введення-виведення, відображене у пам’ять
- Аналогове введення-виведення



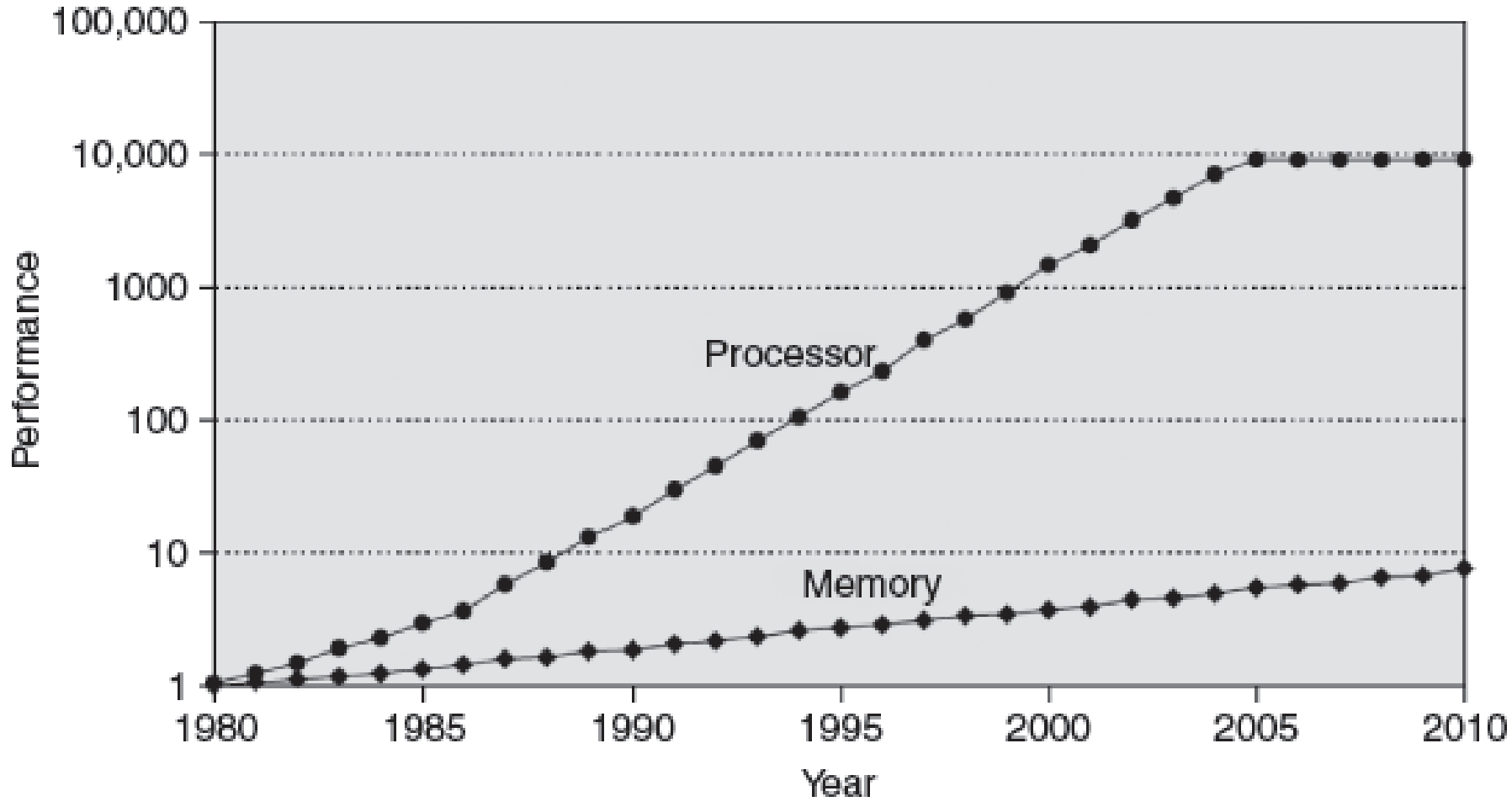
Вступ

- Продуктивність комп’ютера залежить від:
  - продуктивності процесора
  - продуктивності підсистем пам’яті

Інтерфейс пам’яті



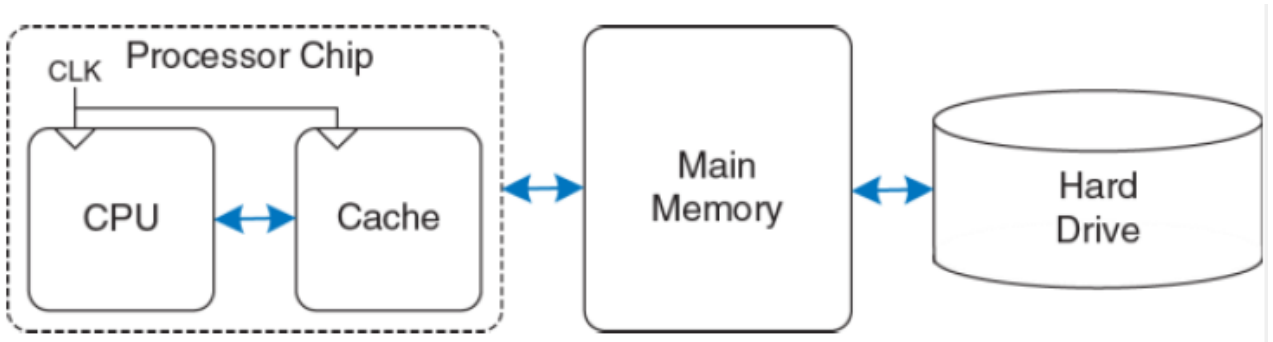
Різниця між продуктивністю процесора і пам’яті



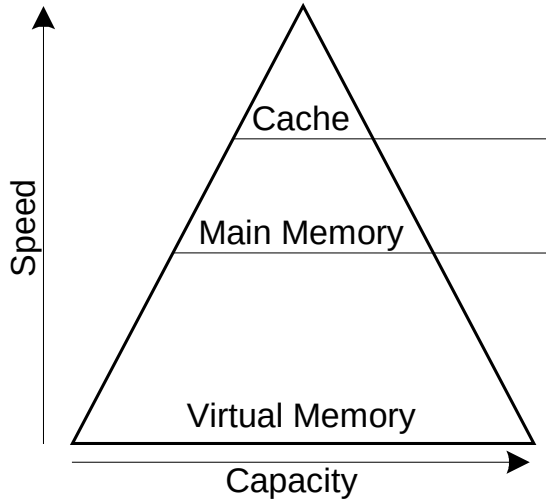
Підвищення продуктивності підсистем пам’яті

- Зробити підсистему пам’яті такою ж швидкою, як і процесор
- Використати ієрархію пам’яті
- Ідеальна пам’ять:
  - швидка
  - дешева (недорога)
  - велика (за обсягом)

Ієрархія пам’яті і підсистеми введення-виведення



Ієрархія пам’яті



Technology	Price / GB	Access Time (ns)	Bandwidth (GB/s)
SRAM	$10,000	1	25+
DRAM	$10	10 - 50	10
SSD	$1	100,000	0.5
HDD	$0.1	10,000,000	0.1

Локальність

Локальність використовується для пришвидшення доступу до пам’яті

- **Часова локальність:**
  - Локальность у часі
  - Якщо дані використовувалися недавно, то ймовірно, що вони скоро знову знадобляться
  - **Як це використовується:** зберігати недавно використовувані дані на більш високих рівнях ієрархії пам’яті
- **Прострова локальність:**
  - Локальність в просторі
  - Якщо дані використовувалися недавно, то ймовірно скоро знадобляться дані розміщені поблизу
  - **Як це використовується:** при доступу до даних переносяться також суміжні дані на більш високі рівні ієрархії пам’яті

Продуктивність пам’яті

- **Попадання:** дані знайдені на цьому рівні ієрархії пам’яті
- **Промахи:** дані не знайдені на цьому рівні ієрархії пам’яті (потрібно перейти на наступний рівень)

Процент попадань = (кількість попадань / кількість доступів до пам’яті) * 100

(100 – процент промахів)

Процент промахів = (кількість промахів / кількість доступів до пам’яті) *100

(100 – процент попадань)

- **Середній час доступа (англ. Average memory access time, AMAT):** середній час, який процесор витрачає на доступ до пам’яті

AMAT = $t_{\text{cache}} + MR_{\text{cache}}[t_{MM} + MR_{MM}(t_{VM})]$,

де t_{cache}, t_{MM} , t_{VM} - час доступу до кешу, оперативної пам’яті і жорсткого диску

Продуктивність пам’яті. Приклад 1

- Програма має 2000 операцій завантажень і зберігань
- 1250 з них знайшли дані у кеш-пам’яті
- Решта даних знаходяться на інших рівнях ієрархії пам’яті
- Коефіцієнт промахів і попадань в кеш-пам’ять:

  - коефіцієнт попадань = 1250/2000 **= 0.625**

  - коефіцієнт промахів = 750/2000 = **0.375** = 1 – коеф. попадань

Продуктивність пам’яті. Приклад 2

- Припустимо, що процесор має 2 рівня ієрархії: кеш-пам’ять і оперативну пам’ять

  t_{cache} = 1 цикл, t_{MM} = 100 циклів

- Середній час доступу для програми з прикладу 1:

  AMAT = $t_{\text{cache}} + MR_{\text{cache}}(t_{MM})$ = [1 + 0.375(100)] циклів = **38.5 циклів**

Кеш пам’ять

- В кеш пам’яті зберігаються часто використовувані дані з оперативної пам’яті
- Має найвищий рівень в ієрархії пам’яті
- Швидка (звичайно час доступа ≈ 1 такт)

Термінологія кеш пам’яті

- **Блок кешу:** слово і декілька сусідніх до нього слів прочитанах з пам’яті
- **Місткість (*C*):** кількість блоків, які можуть помістится в кеш-пам’ять
- **Розмір блоку кеша (*b*):** кількість слів у блоці кешу
- **Кількість блоків (*B*):** кількість блоків в кеш-пам’яті: $B = C/b$
- **Набір блоків (*S*):** набір може містити один або декілька блоків даних
- **Ступінь асоціативності (*N*):** кількість блоків в наборі
- **Кількість наборів (*S*):** кожна адреса пам’яті відображається тільки в один набір кеш пам’яті $S = B/N$

Класифікація кеш-пам’яті

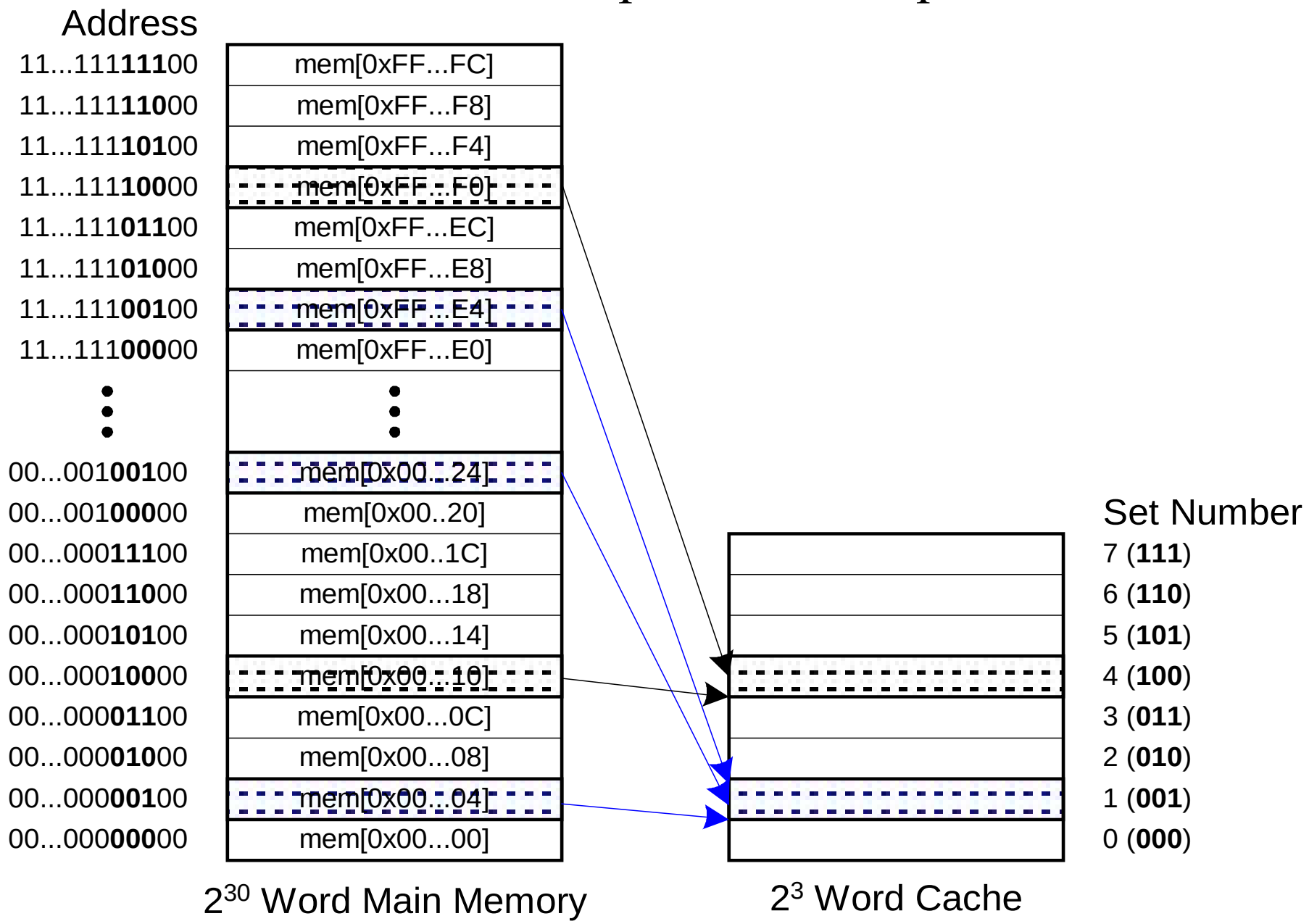
Кеш пам’ять класифікують за числом блоків у наборі S:

- Кеш пам’ять прямого відображення (S=B)
- Кеш пам’ять набірно-асоціативна з N блоками в наборі (S=B/N)
- Кеш пам’ять повністю асоціативна (всі блоки в одному наборі S=1)

Кеш пам’ять прямого відображення

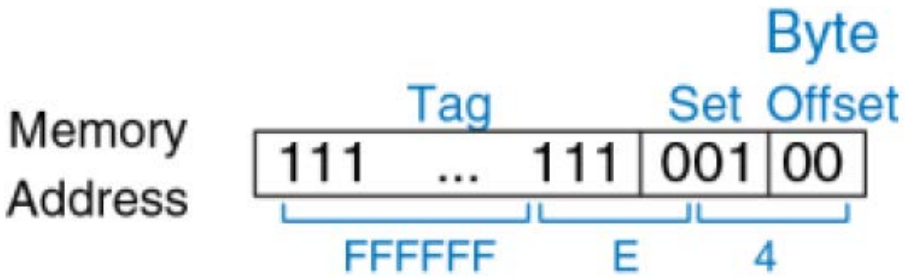
Відображення оперативної 32-розрядної пам’яті на кеш пам’яті місткістю 8 слів, кожне з яких містить машинне32-розрядне слово показано в наступному слайді. Так як адреси оперативної пам’яті вирівняні на межу слів, то молодші два біти адреси завжди будуть нульовими. Наступні $log_2 8 = 3$ біти адресують один із восьми наборів кеш пам’яті, в який буде відображена ця оперативна пам’ять. Тому дані із адрес 0x0000_0004, 0x0000_00024, …, 0xFFFF_FFF4 будуть відображатися в набір 1 кеш пам’яті. Аналогічно дані із адрес 0x0000_00010, …, 0xFFFF_FFFF0 відображаються в набір 4, і так далі.

Кеш пам’ять прямого відображення

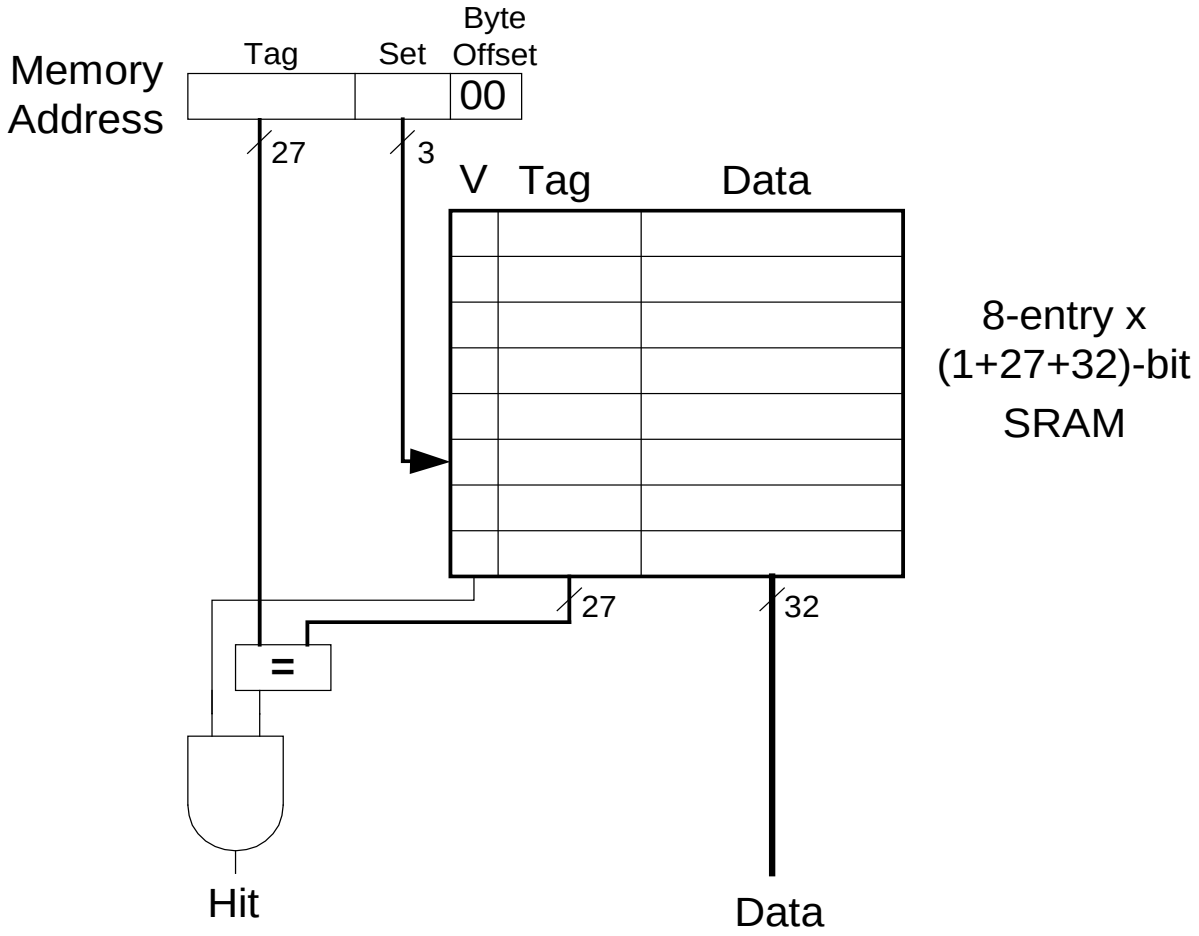


Адресація і апаратна реалізація кеш пам’яті прямого відображення

Так як у кеш пам’ять відображається множина адрес оперативної пам’яті, то потрібно відслідковувати адреси даних, які знаходяться в кожному наборі у поточний момент. Частини адреси при відображенні у кеш пам’ять:



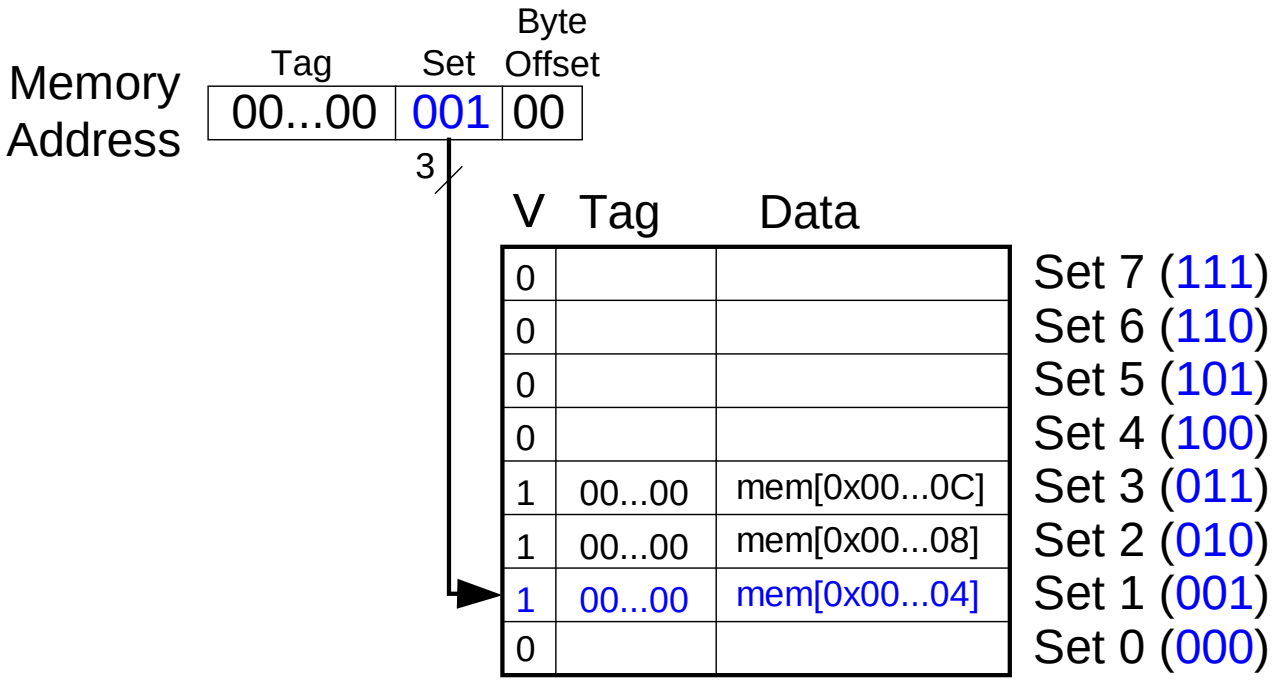
Апаратна реалізація кеш пам’яті



Data – 32-біти даних
Tag – 27-бітів тегу (адресація ОП)
V – біт достовірності

Продуктивність кешу прямого відображення

```
# MIPS код
        addi $t0, $0, 5
loop:   beq  $t0, $0, done
        lw   $t1, 0x4($0)
        lw   $t2, 0xC($0)
        lw   $t3, 0x8($0)
        addi $t0, $t0, -1
        j    loop
done:
```



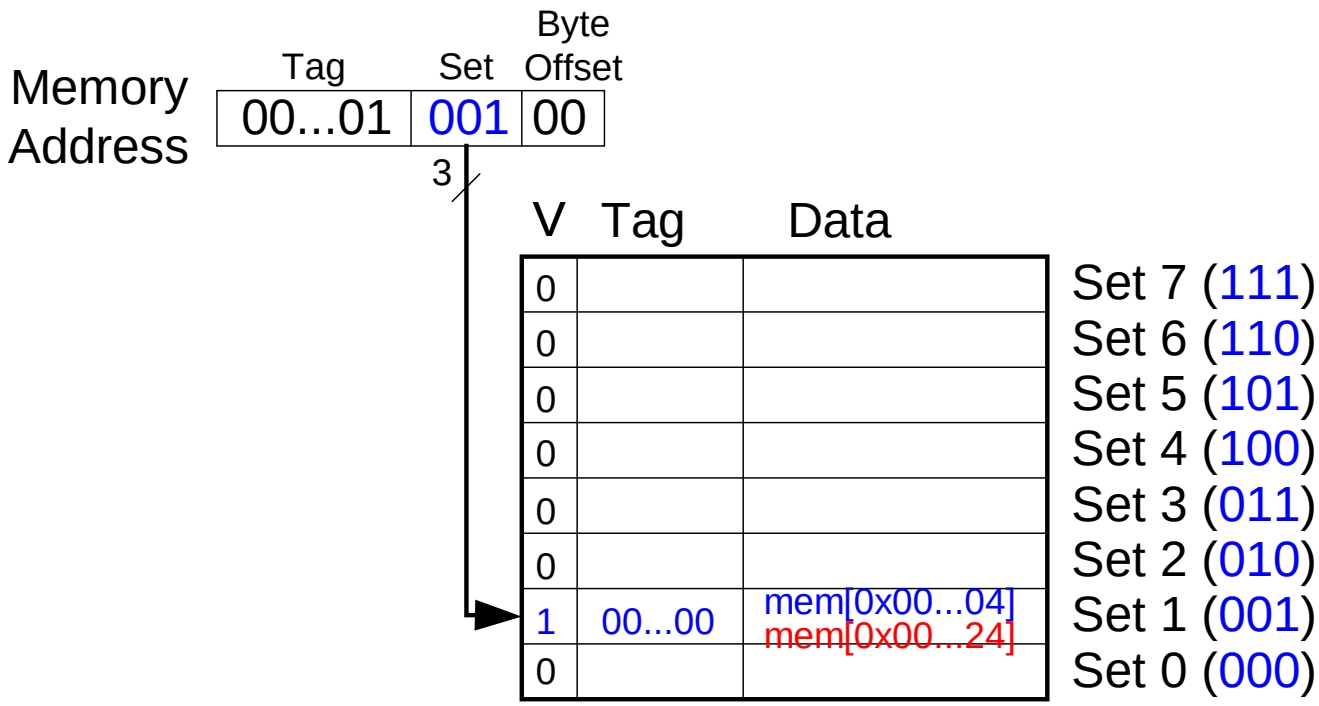
На кожному циклі три звернення до пам’яті (команда lw).

На першому циклі кеш пам’ять порожня. Процент промахів 3/15 = 20%

Часова локальність. Обов’язкові промахи

Кеш прямого відображення: конфлікти

```
# MIPS код
        addi $t0, $0, 5
loop:   beq  $t0, $0, done
        lw   $t1, 0x4($0)
        lw   $t2, 0x24($0)
        addi $t0, $t0, -1
        j    loop
done:
```

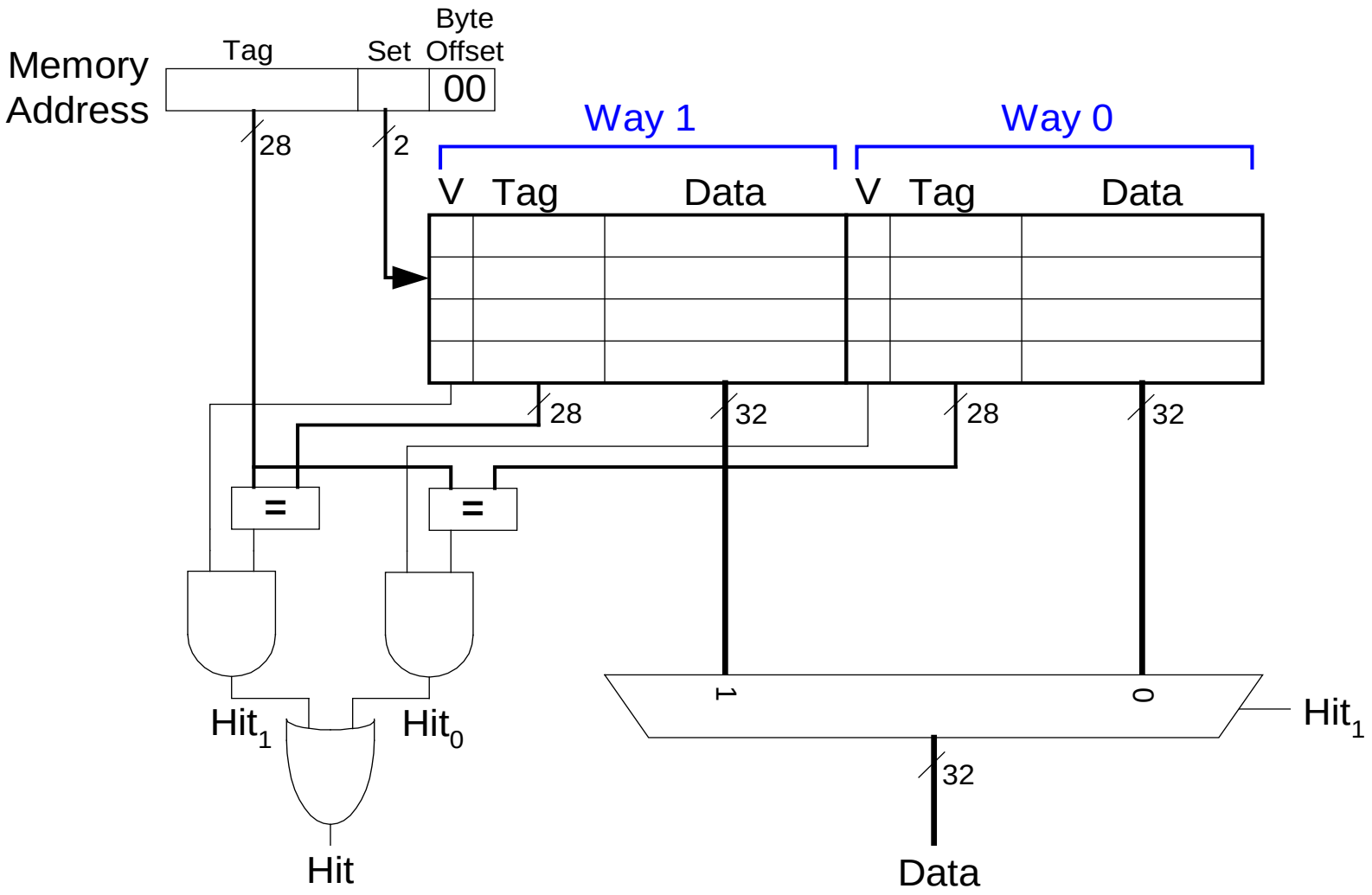


На кожному циклі два звернення до пам’яті (команда lw).

Обидві адреси пам’яті 0x04 і 0x24 відображаються в один набір.

На кожній ітерації значення в наборі 1 перезаписується значенням комірок з адрес 0x04 і 0x24. Ці дві адреси конфліктують, так що процент помахів кешу пам’яті 100%.

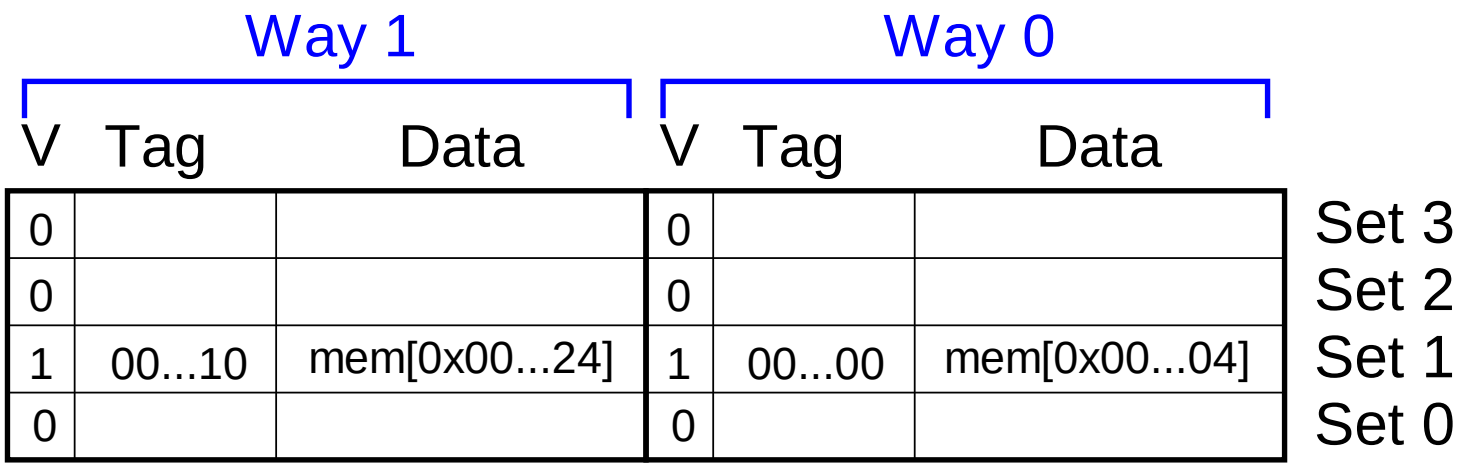
N-секційна набірно-асоціативна кеш пам’ять



У наборах розширюються кількість N блоків (секцій) для зменшення числа конфліктів. *N* – ступінь асоціативності. Так кеш має 4 набори замість 8

Продуктивність набірно-асоціативної кеш пам’яті

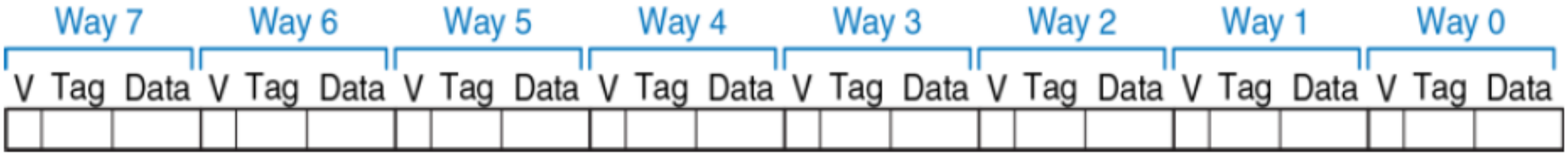
```
# MIPS код
        addi $t0, $0, 5
loop:   beq  $t0, $0, done
        lw   $t1, 0x4($0)
        lw   $t2, 0x24($0)
        addi $t0, $t0, -1
        j    loop
done:
```



Асоціативний кеш місткістю 8 слів і 4 набори по два блоки (секції). Для вибору кожного набору використовується $log_2 4 = 2$ біти, не 3 біти і відповідно розмір тегу збільшується з 27 до 28 бітів, що дозволяє записувати дані в один або інший блок. За рахунок цього зменшується процент промахів через конфлікти до 2/10 = 20% .

Повністю асоціативна кеш пам’ять

Складається тільки з одного набору і *B* блоків (секцій). Адреса пам’яті може бути відображена в любий із цих блоків.

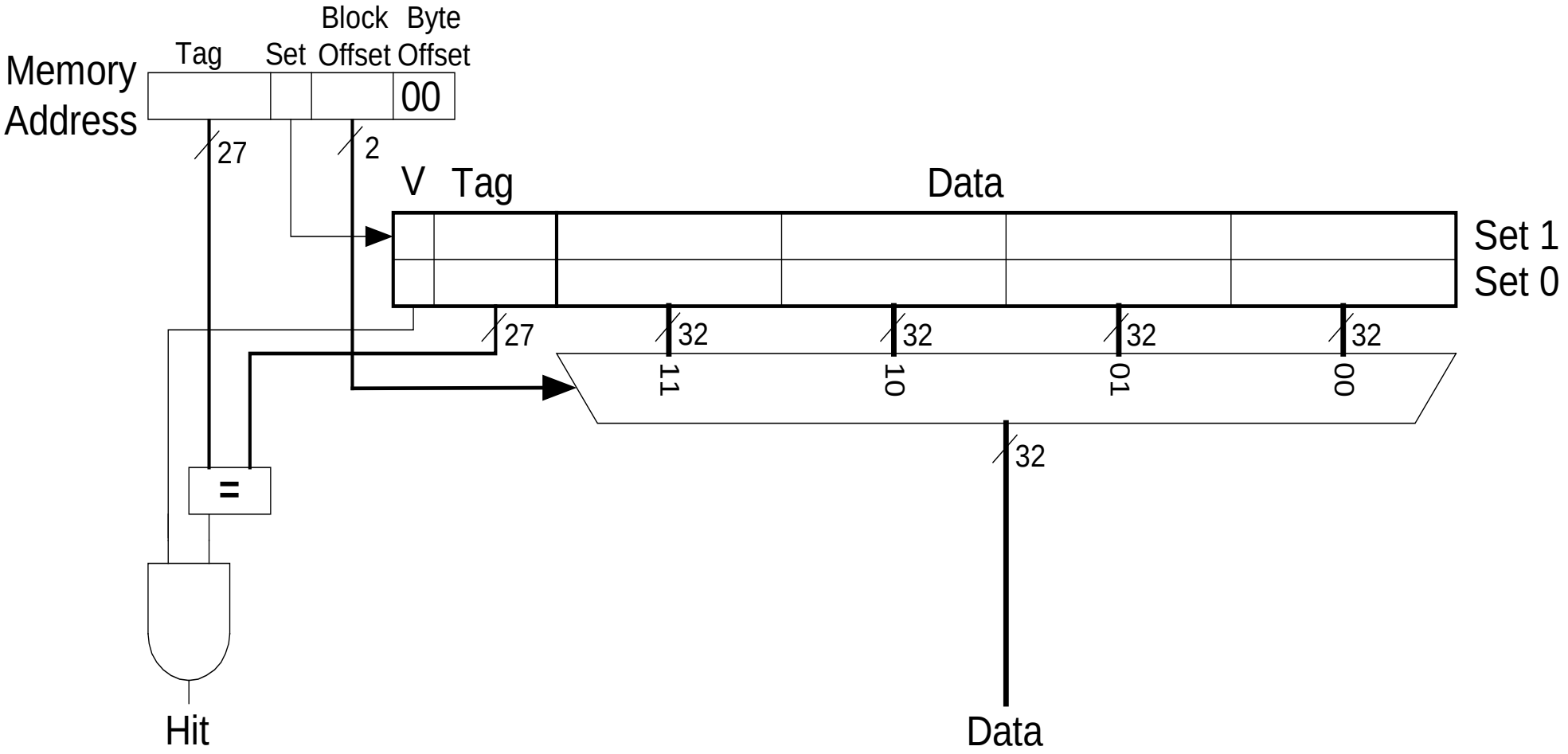


Зменшує кількість конфліктів через промахи.

Апаратна реалізація дуже витратна.

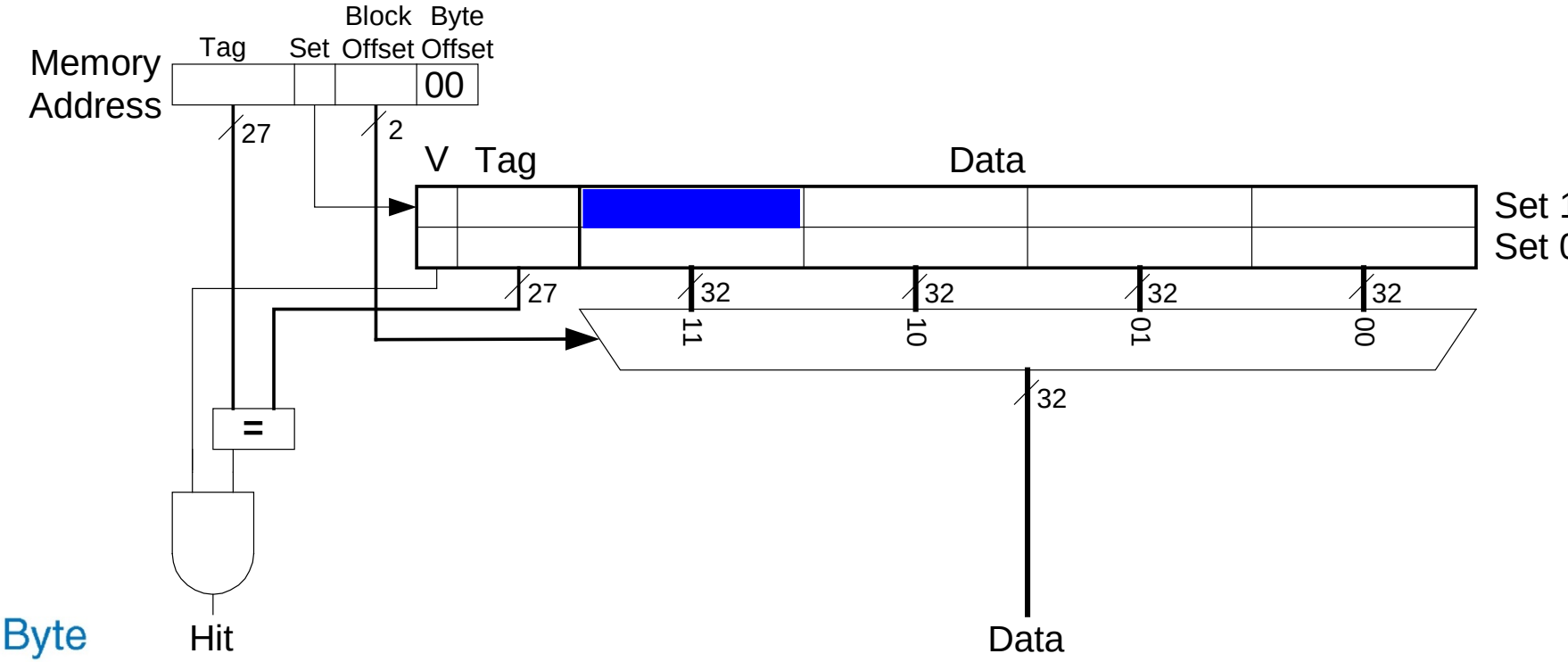
Просторова локальність

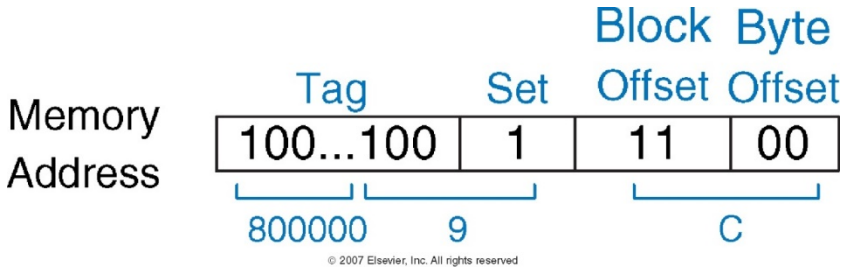
Для використання просторової локальність застосовують великі за розміром блоки, в які записуються крім прочитаного слова з ОП і декілька його сусідніх слів. Розмір блоку b=4, кількість слів C=8. Пряме відображення - один блок один набір. Кількість блоків B=C/2=8/4=2.



Кеш пам’ять з більшим розміром рядка

```
# MIPS код
        addi $t0, $0, 5
loop:   beq  $t0, $0, done
        lw   $t1, 0x4($0)
        lw   $t2, 0x24($0)
        addi $t0, $t0, -1
        j    loop
done:
```





Процент промахів 1/15 = 6.67%. Кеш пам’ять з більшим розміром блоку зменшує промахи за рахунок просторової локальності

Заміна у кеші рідко використовуваних даних

Використовуючи принцип часової локальності у кешах застосовується стратегія заміни рідко використовуваних даних (англ., least recently used, LRU). У 2-блоковій набірно-асоціативній кеш пам’яті є біт використання U (від англ. used), який містить номер того блоку, який довше не використовувався. Кожний раз, коли відбувається доступ до блоку, біт U встановлюється таким чином, щоб вказувати на інший блок.

U=0 дані в секції 0 довше не використовувалися ніж в секції 1

```
lw $t0, 0x04($0)
lw $t1, 0x24($0)
```

	Way 1				Way 0			
	V	U	Tag	Data	V	Tag	Data	
Set 3 (11)	0	0			0			
Set 2 (10)	0	0			0			
Set 1 (01)	1	0	00...010	mem[0x00...24]	1	00...000	mem[0x00...04]	
Set 0 (00)	0	0			0			

(a)

Заміни даних в блоці 0 і U=1 дані в секції 1 довше не використовувалися ніж в секції 0

```
lw $t2, 0x54($0)
```

	Way 1				Way 0			
	V	U	Tag	Data	V	Tag	Data	
Set 3 (11)	0	0			0			
Set 2 (10)	0	0			0			
Set 1 (01)	1	1	00...010	mem[0x00...24]	1	00...101	mem[0x00...54]	
Set 0 (00)	0	0			0			

(b)

Висновки організації кеш пам’яті

- Кеш є двовимірний масив, рядки якого називають наборами, в стовпці – блоками (секціями)
- Кожний елемент масиву містить блок даних і зв’язаний із ним тег та біт достовірності.
- Кеш характеризується:
  - Місткістю C
  - довжиною блоку b і кількістю блоків B=C/b
  - числом блоків у наборі

Способи організації кеш пам’яті

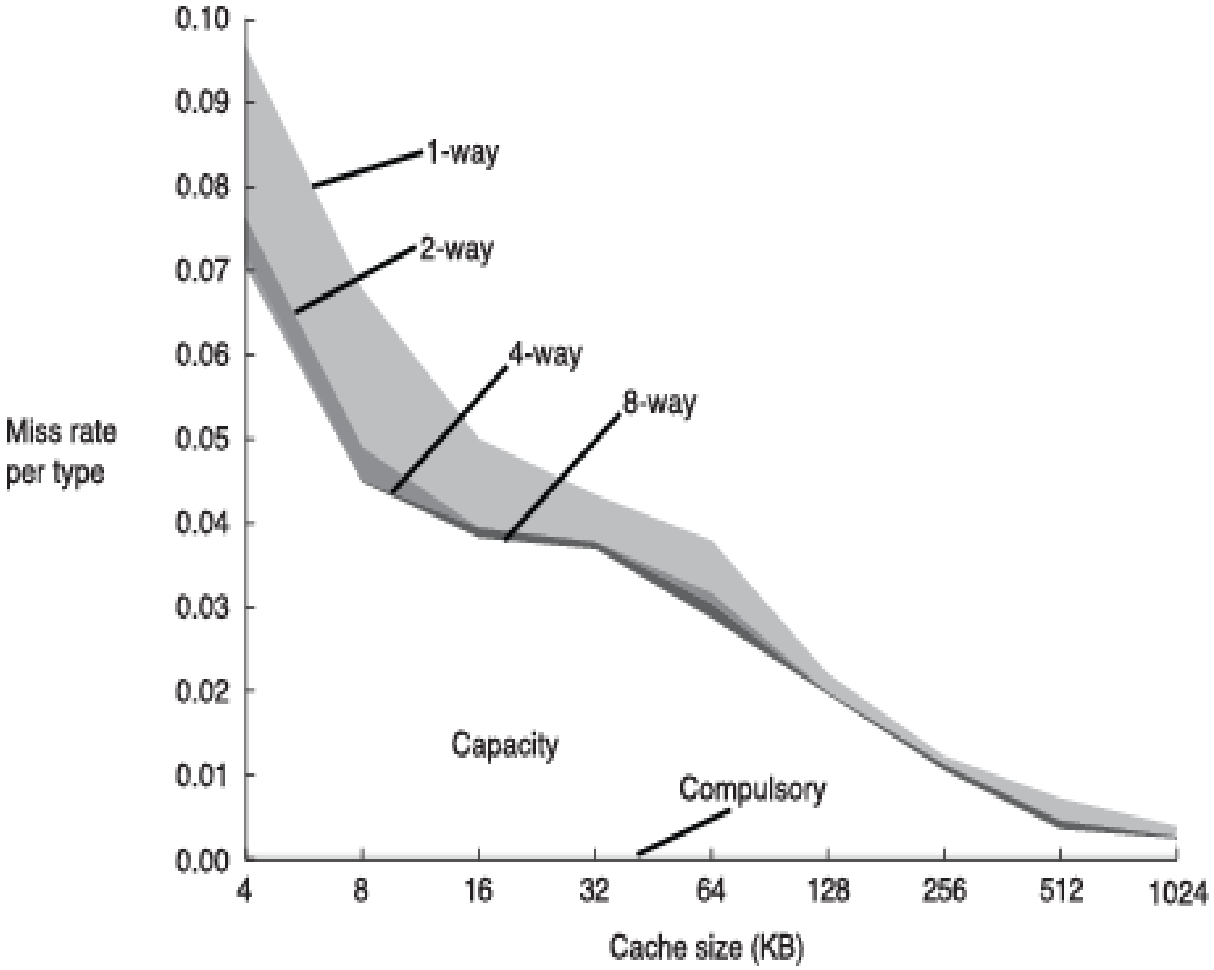
Спосіб організації	Кількість блоків (N)	Кількість наборів ($S = B/N$)
Прямого відображення	1	B
Набірно-асоціативний	$1 < N < B$	B / N
Повністю асоціативний	B	1

Типи промахів

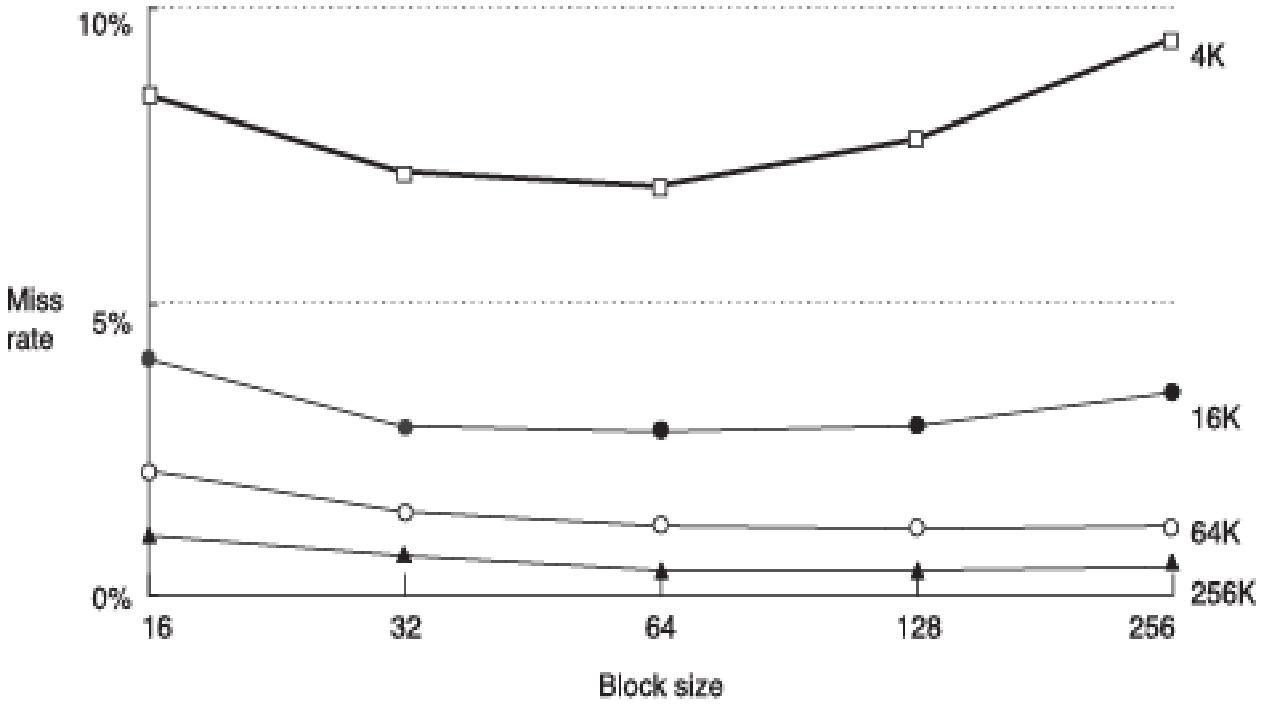
- **Неминучі**: при першому доступі до даних
- **Через недостатню місткість:** кеш пам’ять занадто мала, щоб вмістити зразу всі потрібні дані
- **Через конфлікти:** дані відображаються в один і той же набір кеш пам’яті
- **Ціна промаху:** час, потрібний для видобування рядка з більш низкого рівня ієрархії
- **Заміна рідко використовуваних даних (англ. Least recently used, LRU)**: витіснення того рядка, який довше всього не використовувався

Динаміка процентів промахів

В залежності від кількості блоків в наборі

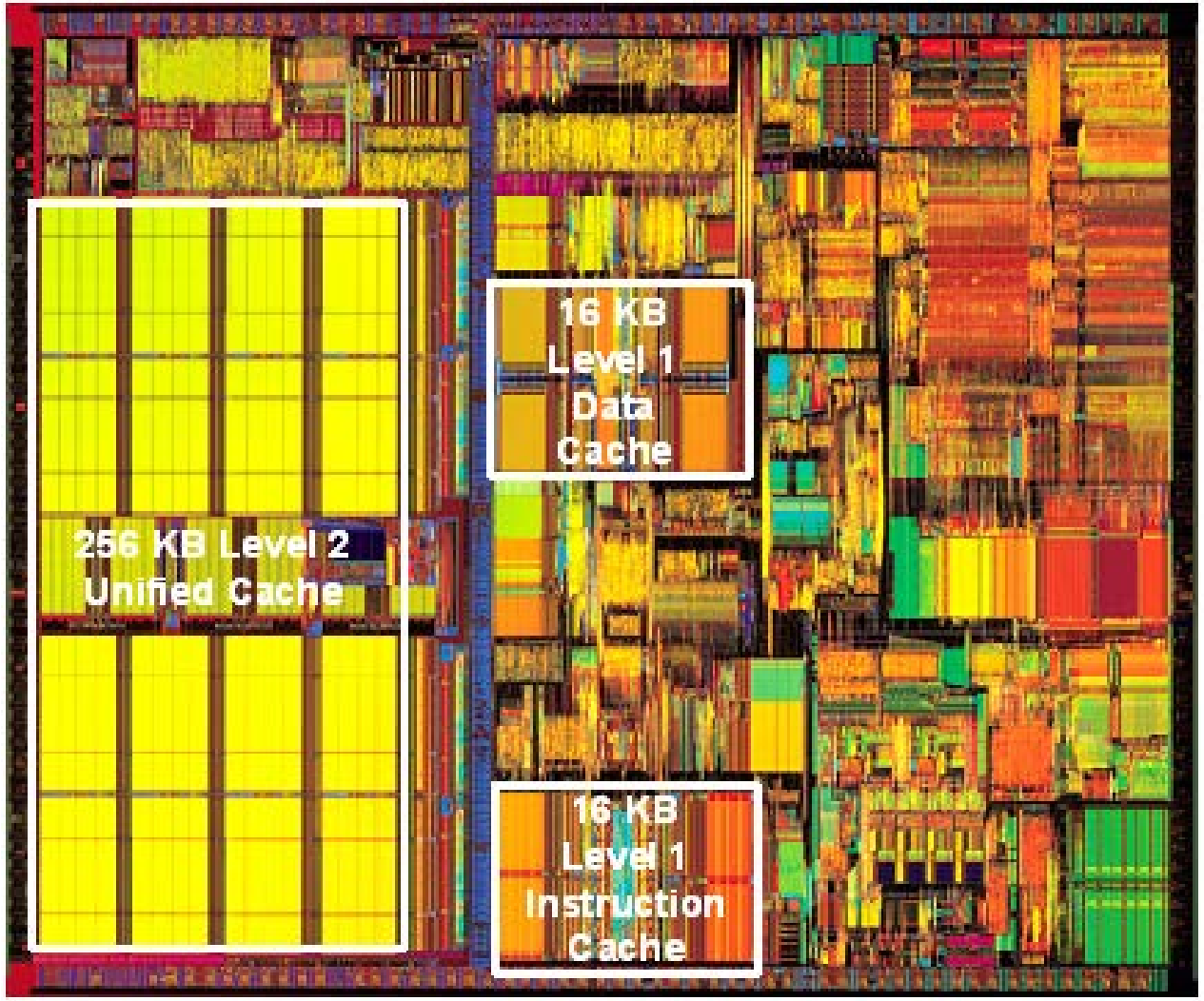


В залежності від розміру блоку

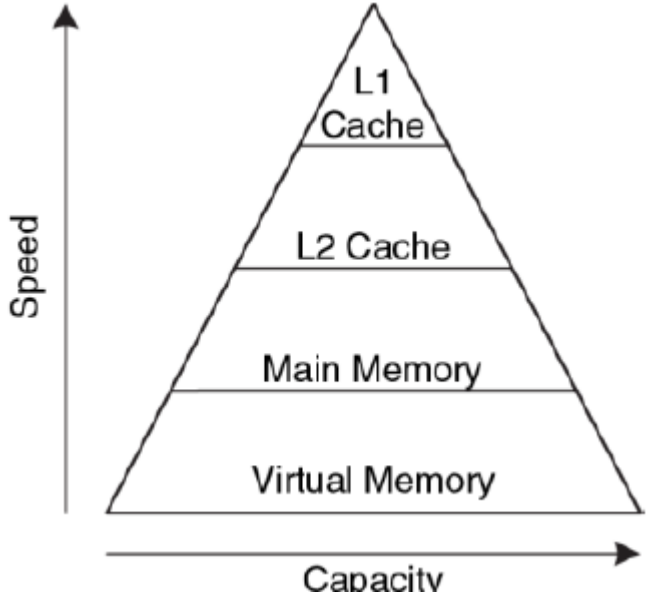


Багаторівневі кеші

Intel Pentim III

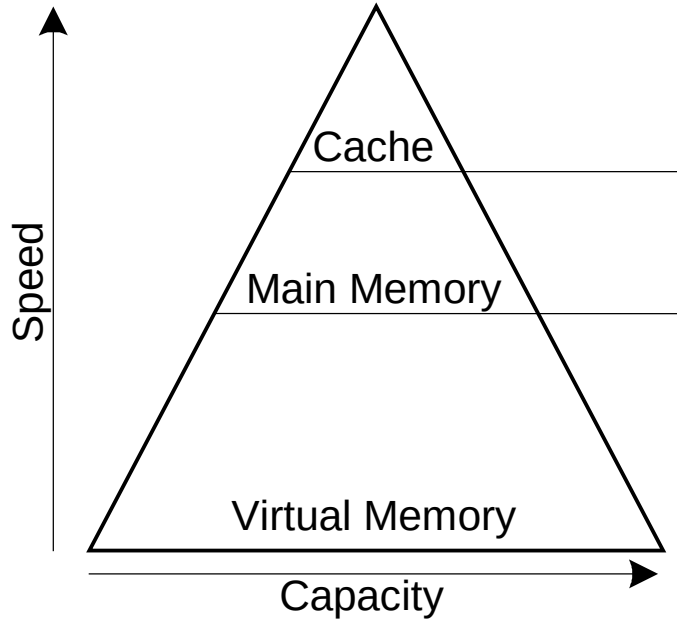


- Кеші більшого розміру мають менший процент промахів, але більш тривалий час доступу
- Рівень кешу 1 (L1): маленький і швидкий (наприклад 16 KB, 1 такт)
- Рівень кешу 2 (L2): більший і повільніший (наприклад 256 KB, 2-6 циклів)
- Більшість сучасних комп’ютерів мають кеші L1, L2 і L3



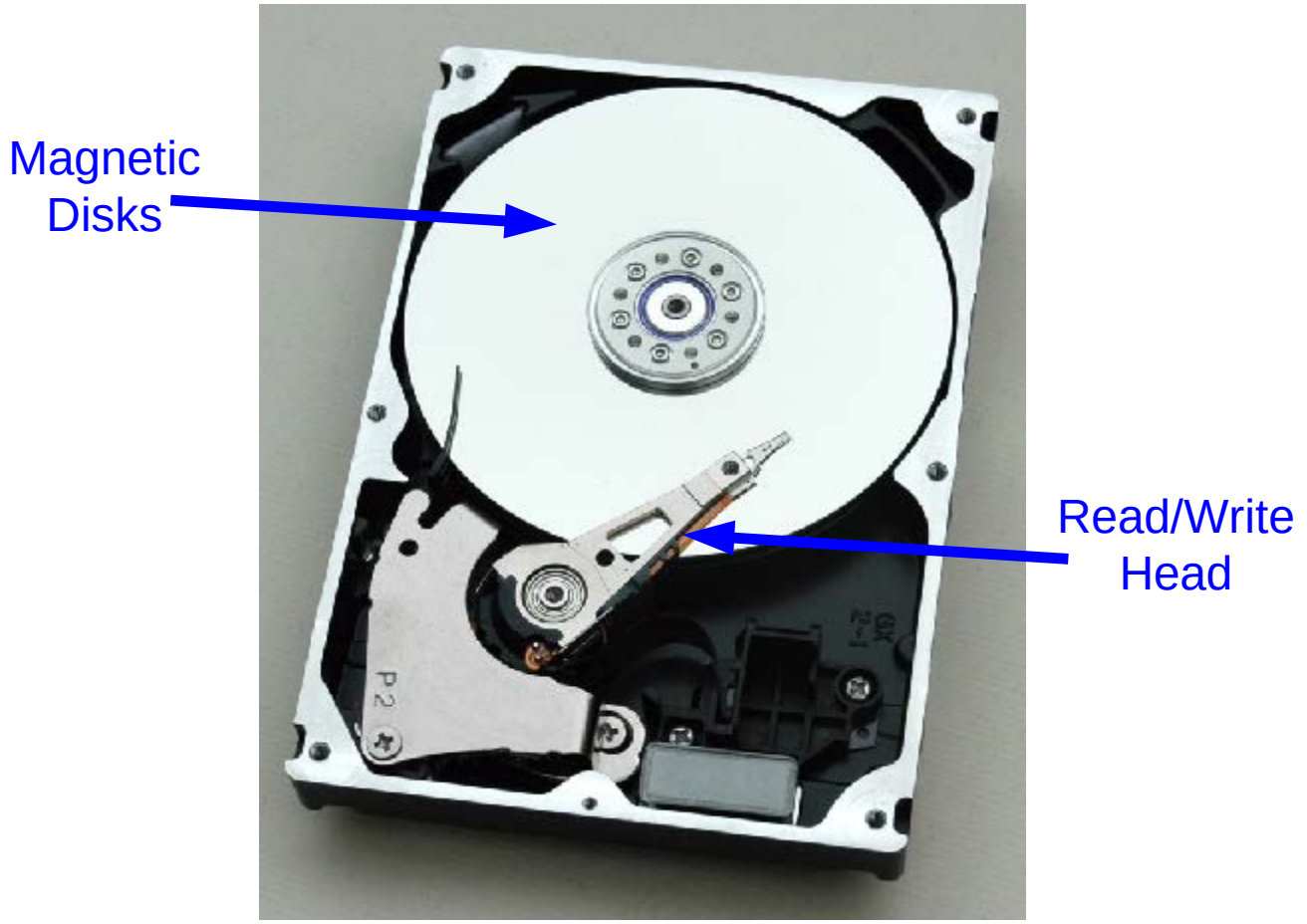
Віртуальна пам’ять

- Віртуальна пам’ять (жорсткий диск) створює ілюзію великого розміру пам’яті
  - повільна, дуже велика, дуже дешева
- Фізична оперативна пам’ять (DRAM) використовується як кеш для жорсткого диску
  - швидка, велика, дешева



Technology	Price / GB	Access Time (ns)	Bandwidth (GB/s)
SRAM	$10,000	1	25+
DRAM	$10	10 - 50	10
SSD	$1	100,000	0.5
HDD	$0.1	10,000,000	0.1

Жорсткий диск



Пошук потрібного блоку даних займає мілісекунди

Віртуальна пам’ять

- **Віртуальні адреси**
  - Програми використовують віртуальні адреси
  - Увесь віртуальний простір зберігається на жорсткому диску
  - Підмножина віртуальних адрес даних зберігається в DRAM
  - Центральний процесор транслює віртуальні адреси в **фізичні адреси** (DRAM адреси)
  - Дані, які не поміщаються в DRAM, вивантажуються на жорсткий диск
- **Захист пам’яті**
  - Кожна програма має свій віртуальний адресний простір, який відображається у фізичний
  - Дві программи можуть використовувати той же віртуальний адрес для різних даних
  - Програми не повинні знати, як працюють інші програми
  - Одна програма (або вірус) не може пошкодити пам’ять, яка використовується іншою програмою

Аналогія між віртуальною і кеш пам’яттю

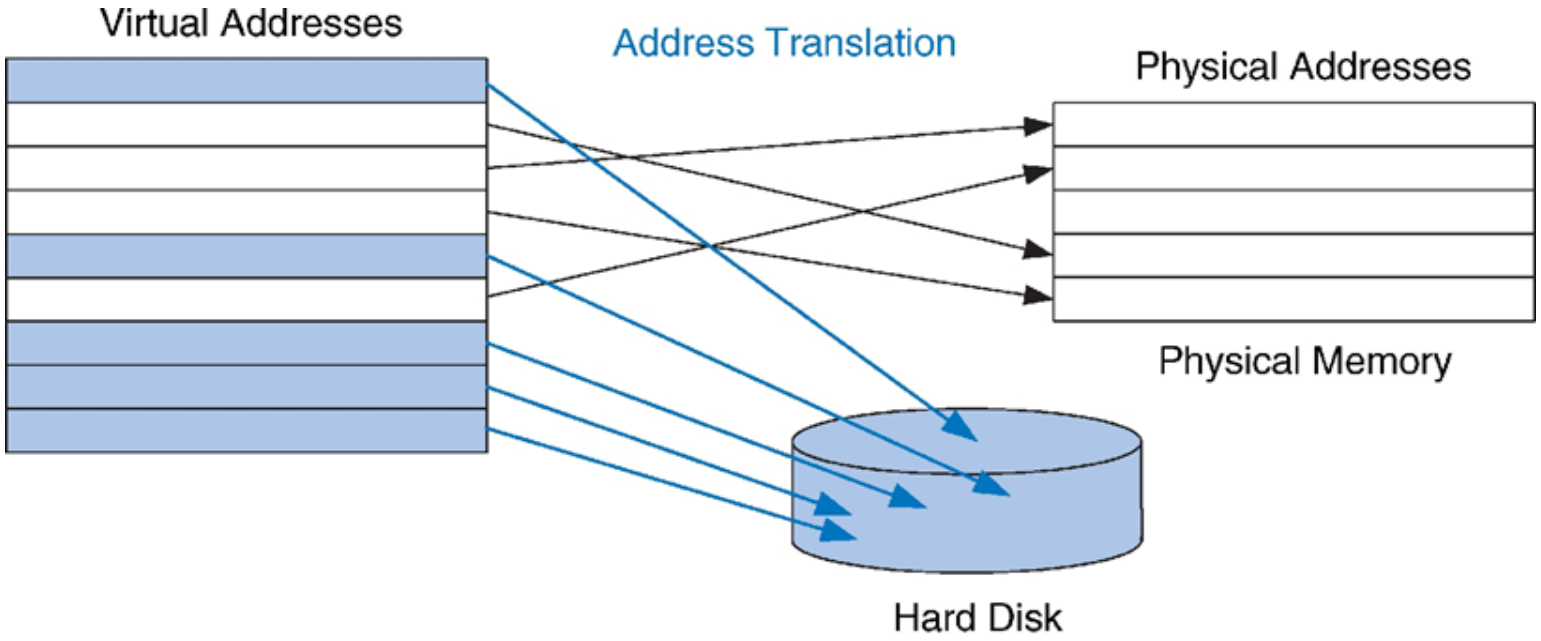
Кеш пам’ять	**Віртуальна пам’ять**
Рядок	Сторінка
Розмір рядка	Розмір сторінки
Зміщення відносно початку рядка	Зміщення відносно початку сторінки
Промах	Сторінкова помилка
Тег	Номер віртуальної сторінки

Фізична пам’ять використовується як кеш віртуальної пам’яті

Термінологія віртуальної пам’яті

- **Розмір сторінки:** кількість пам’яті, яка переноситься з жорсткого диску в DRAM одночасно
- **Трансляція адреси:** визначення фізичної адреси за віртуальною
- **Таблиця сторінок:** таблиця пошуку, яка використовується для трансляції віртуальних адрес у фізичні
- Більшість доступів здійснюється у фізичну пам’ять, але програми мають велику за місткістю віртуальну пам’ять

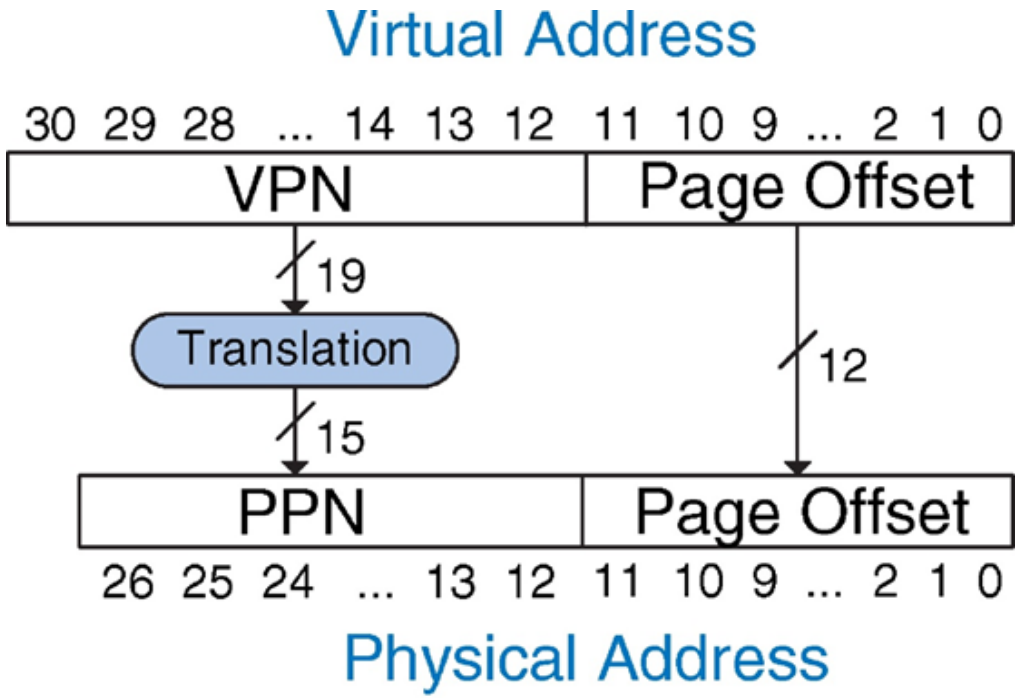
Віртуальні і фізичні адреси



Трансляція адреси

Система:

- Віртуальна адреса: **31** біт
- Розмір віртуальной пам’яті: 2 ГБ = 2^{31} байт
- Фізична адреса: **27** біт
- Розмір фізичної пам’яті: 128 МБ = 2^{27} байт
- Розмір сторінки: 4 КБ = 2^{12} байт
- Кількість віртуальних сторінок (англ. virtual page number, VPN) , VPN=2^{31} / 2^{12} = 2^{19}
- Кількість фізичних сторінок (англ. physical page number, PPN) , PPN=2^{27} / 2^{12} = 2^{15}
- Кількість фізичних сторінок менша від кількості віртуальних сторінок в 2^{19} / 2^{15} = 2^4 = 16 разів



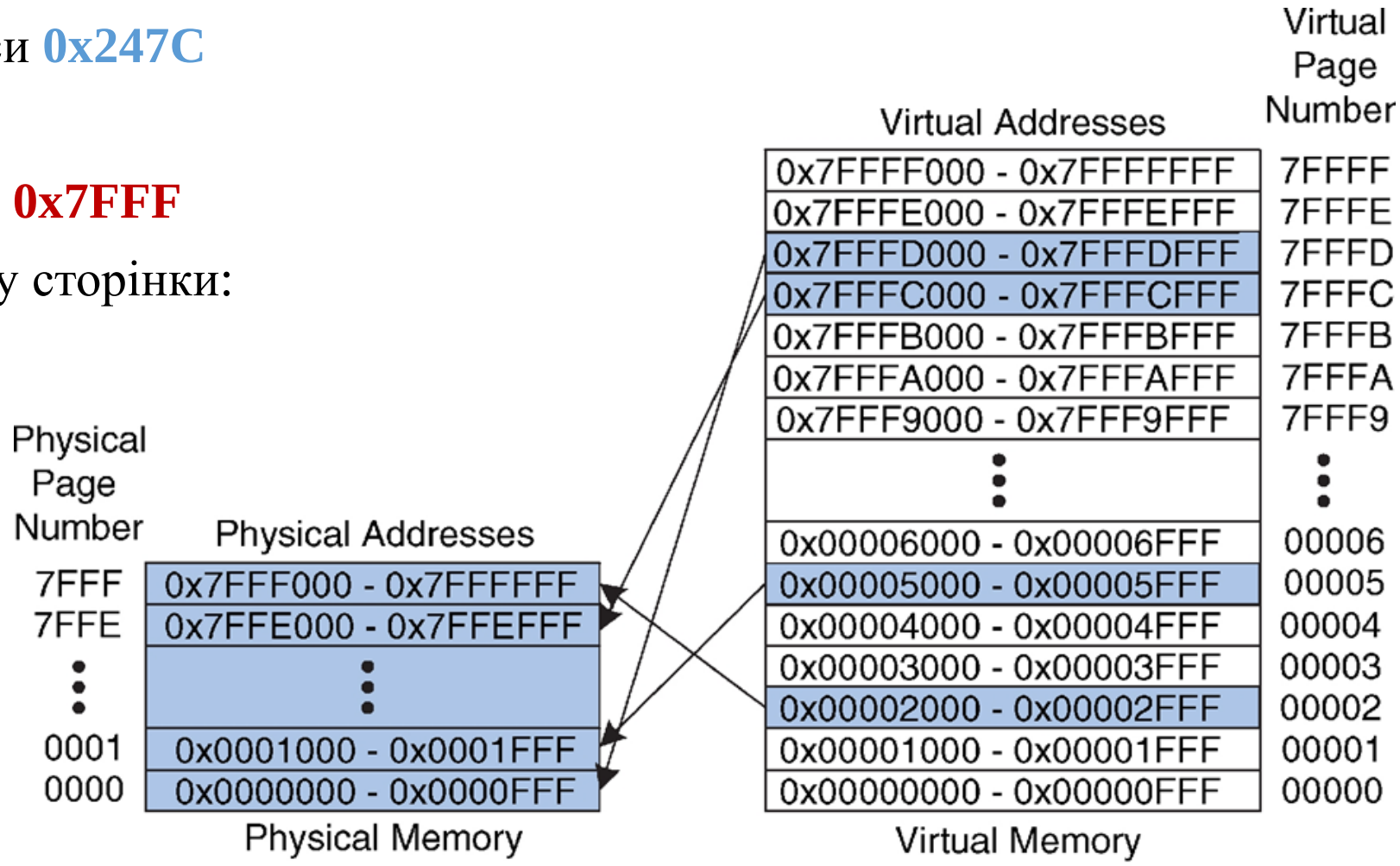
Молодші 12 біт визначають зміщення від початку сторінки і тому не транслюються

Відображення віртуальних сторінок на фізичні

Фізична адреса віртуальної адреси **0x247C**

- VPN = **0x2**
- VPN 0x2 відображається у PPN **0x7FFF**
- 12-бітове зміщенняе від початку сторінки: **0x47C**
- Фізична адреса = **0x7FFF47C**

Віртуальна сторінка	Фізична сторінка
2	7FFF
5	0001
7FFC	7FFE
7FFD	0000

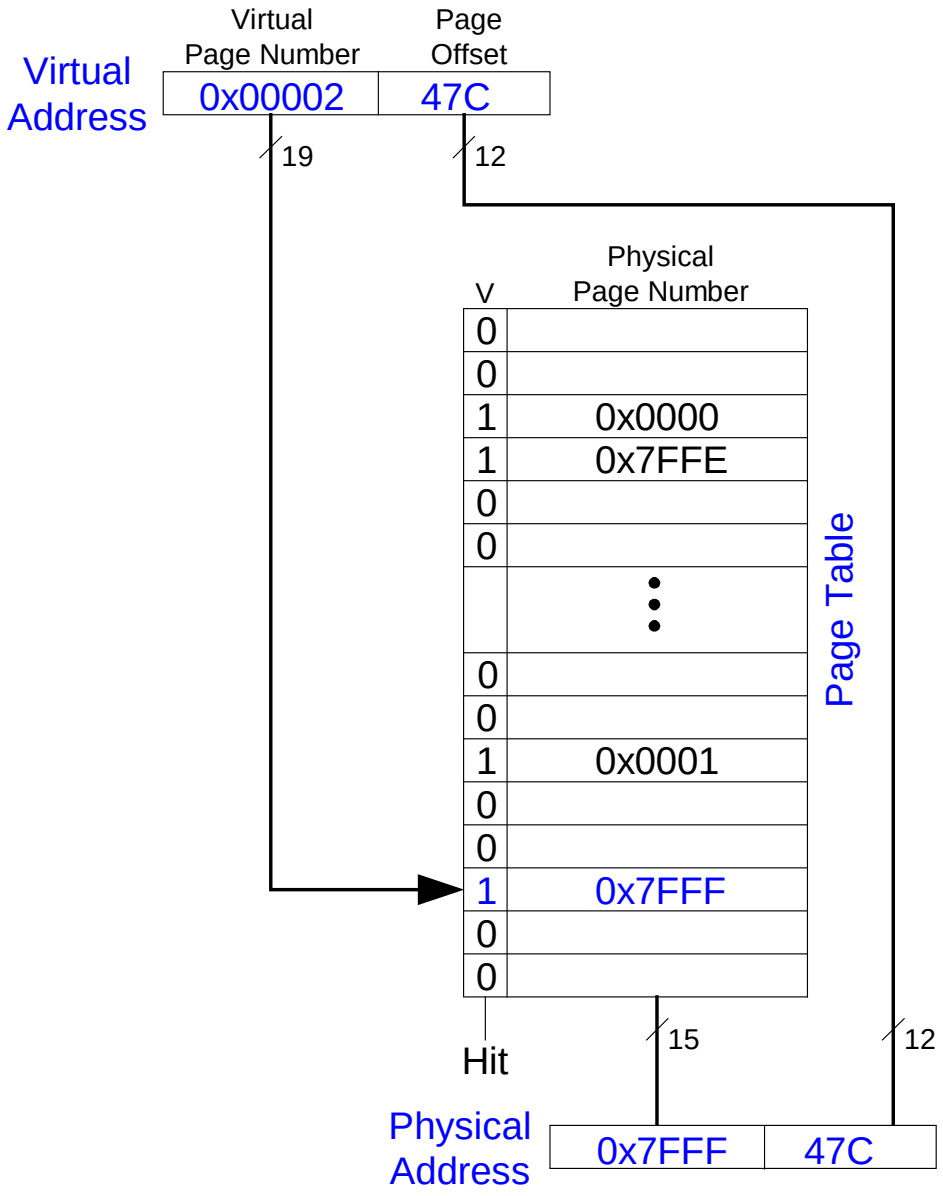


- 19-бітовий номер віртуальної сторінки
- 15-бітовий номер фізичної сторінки

Таблиця сторінок

Таблиця сторінок:

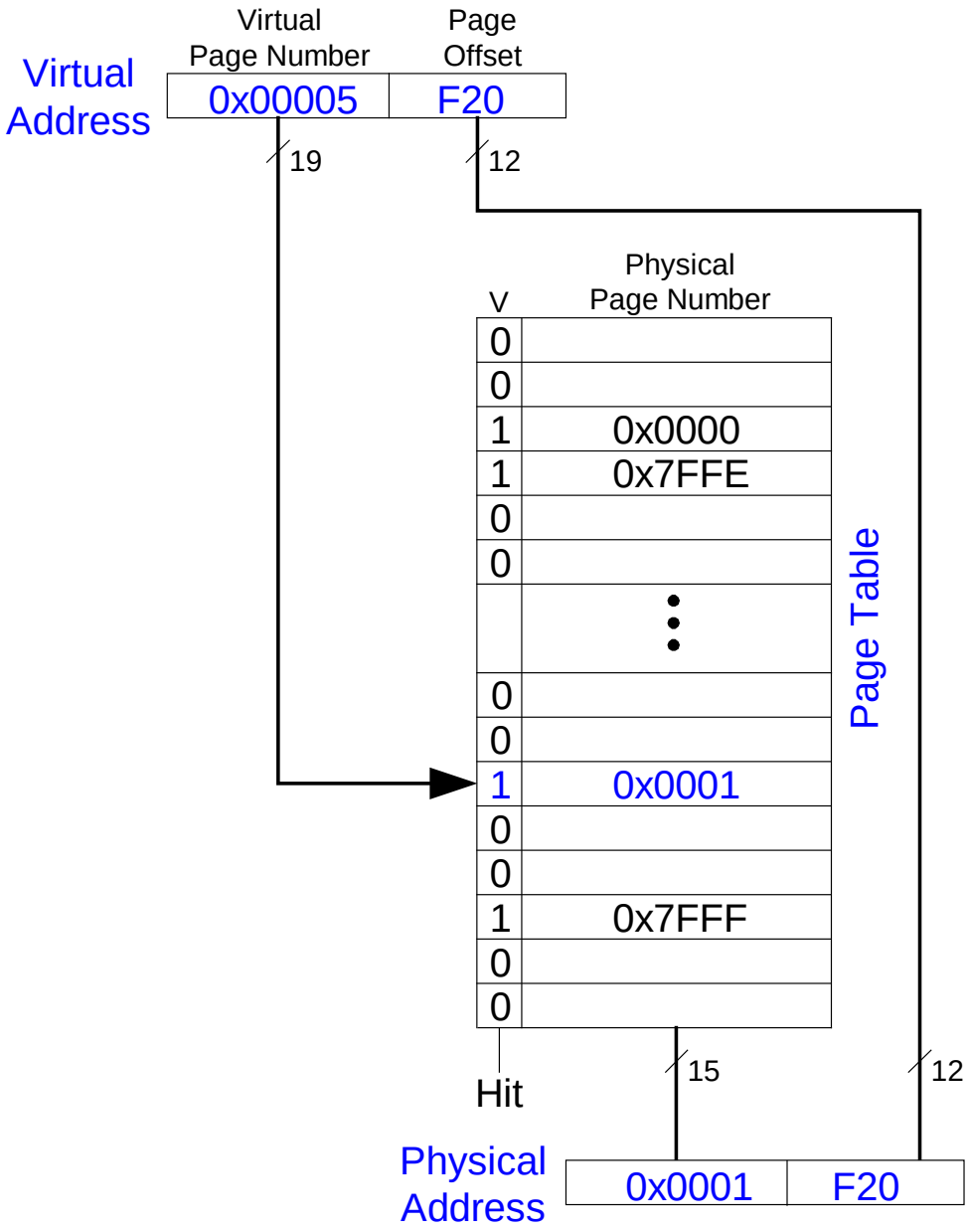
- містить віртуальний номер сторінки (VPN), який є індесом у таблиці сторінок
- містить запис для кожної віртуальної сторінки
- запис містить:
  - **біт достовірності:** 1 якщо сторінка знаходиться у фізичній пам’яті
  - **номер фізичної сторінки:** розміщення сторінки



Таблиця сторінок. Приклад 1

Фізична адреса віртуальної адреси **0x5F20**:

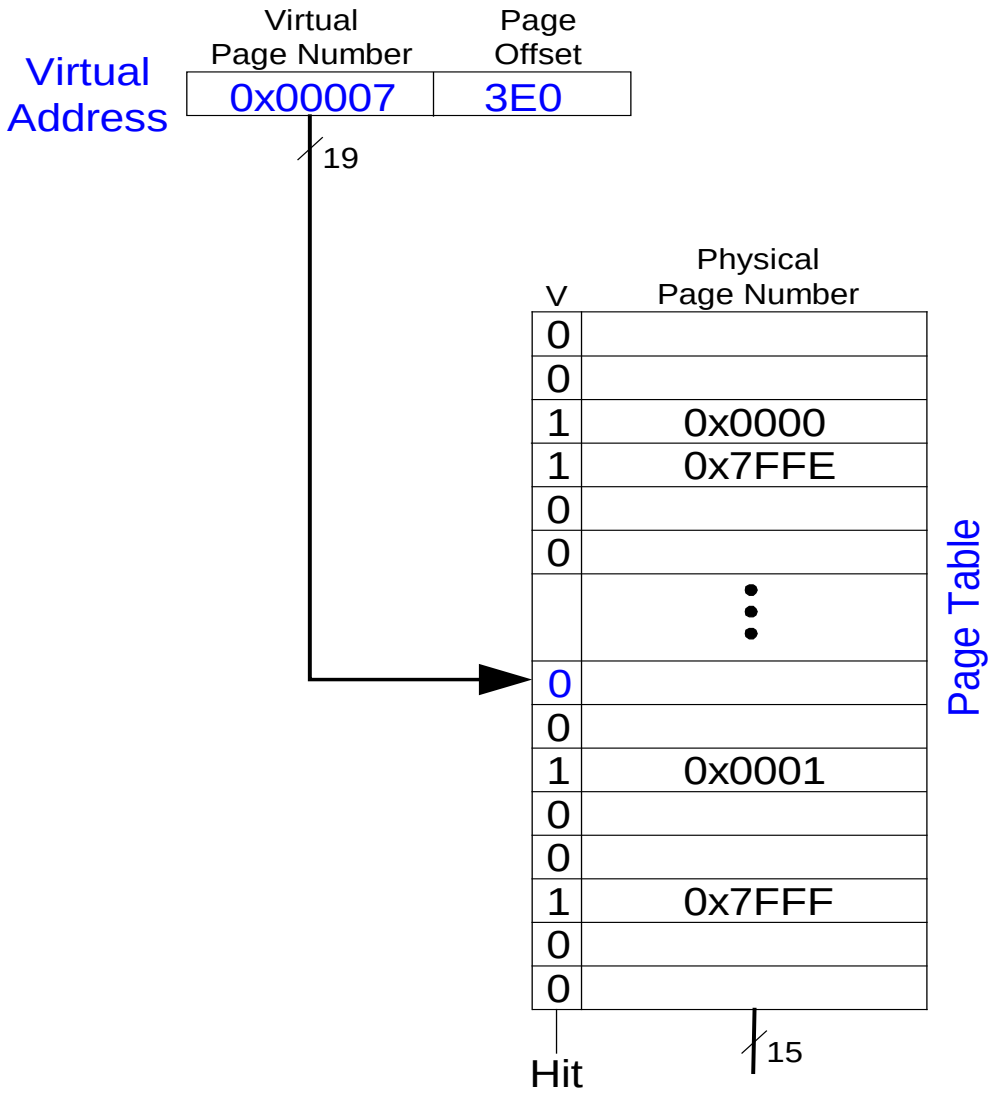
- VPN = **5**
- запис з № 5 у таблиці сторінок VPN 5 вказує на фізичну сторінку **1**
- фізична адреса: **0x0001F20**



Таблиця сторінок. Приклад 2

Фізична адреса віртуальної адреси **0x73E0**:

- VPN = **7**
- Запис 7 є недостовірним
- Віртуальна сторінка має бути ***завантажена*** у фізичну пам’ять з диску



Проблеми таблиці сторінок

- Таблиця сторінок велика
  - звичайно, розміщується у фізичній пам’яті
- Для завантаження/зберігання потрібно два доступи до оперативної пам’яті:
  - один для трансляції (читання з таблиці сторінок)
  - один для доступу до даних (після трансляції)
- Зменшує продуктивність пам’яті в 2 рази

Буфер асоціативної трансляції (з англ. Translation lookside buffer, TLB)

- Невеликий кеш самих останніх трансляцій
- Зниження кількості доступів до пам’яті для більшості завантажень/зберігань з 2 до 1

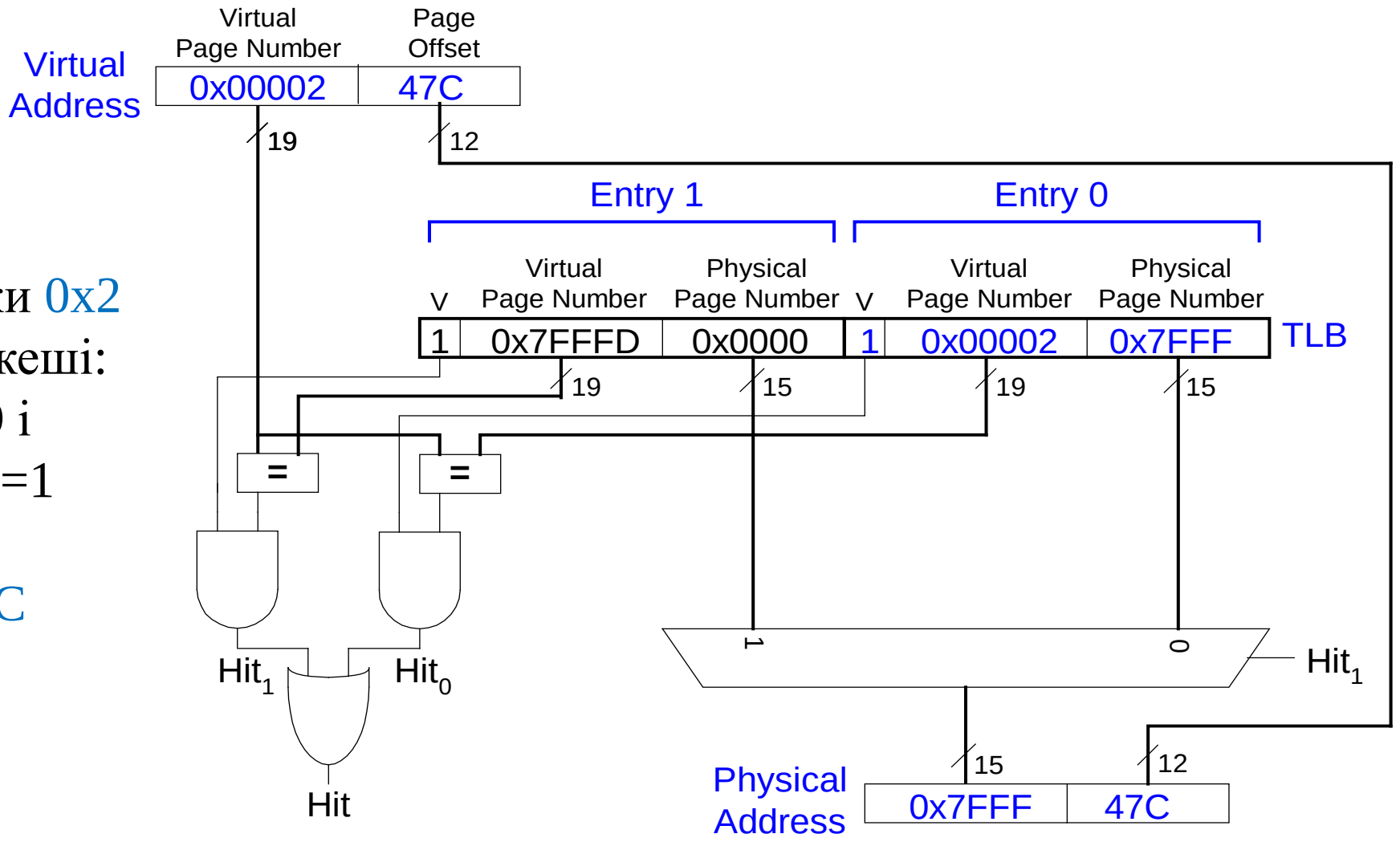
Буфер асоціативної трансляції, TLB

- Доступ до таблиці сторінок: велика просторова локальність
  - Великий розмір сторінки: завантаження/зберігання, які йдуть один за одним мають велику ймовірність доступу до однієї і тієї ж сторінки
- TLB
  - Невеликий: доступ < 1 такту
  - Звичайно містить 16 – 512 запитів
  - Повністю асоціативний
  - Звичайно процент попадань > 99 %
  - Зниження кількості доступів до пам’яті для більшості завантажень/зберігань з 2 до 1

Трансляція адреси з використанням TLB

Віртуальна адреса 0x247C
Номер віртуальної сторінки 0x2
Порівняння з номерами у кеші:

- співпадіння для блоку 0 і запис дійсний, так як V=1
- фізична адреса після трансляції 0x7FFF_047C



Захист пам’яті

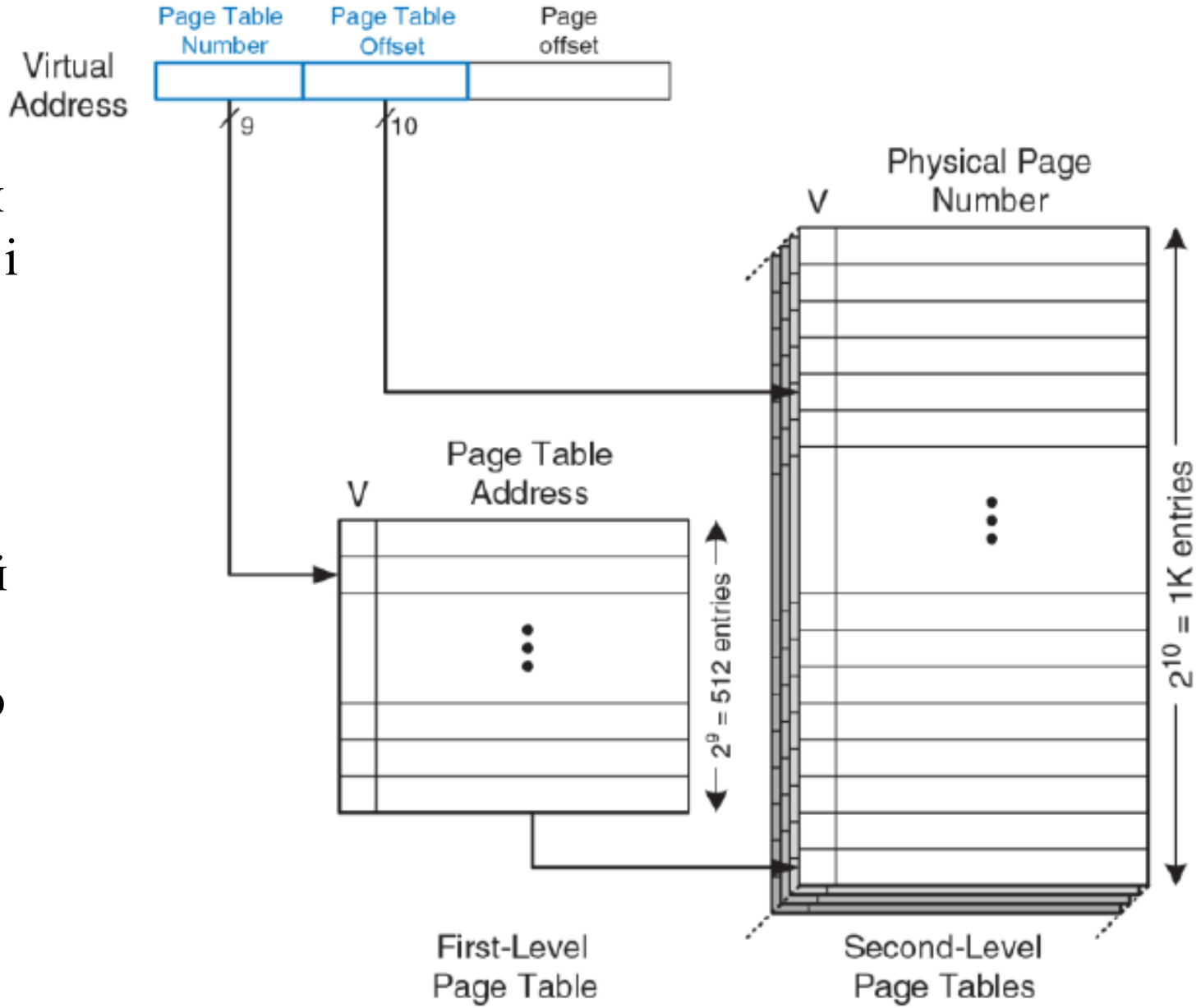
- Множина процесів (програм) працює одночасно
- Кожний процес має свою власну таблицю сторінок
- Кожний процес може використовувати увесь віртуальний адресний простір
- Процес може отримути доступ тільки до фізичної сторінки, відображеної у його таблиці сторінок

Стратегії заміщення сторінок

- Зворотнього запису (write-back) – фізичка сторінка записується назад на жорсткий диск, тільки при витісненні
- Витіснення рідко використовуваних сторінок (least recently used)
- Наскрізного запису (write-through) – кожний запис у фізичну сторінку призводить і до запису на жорский диск

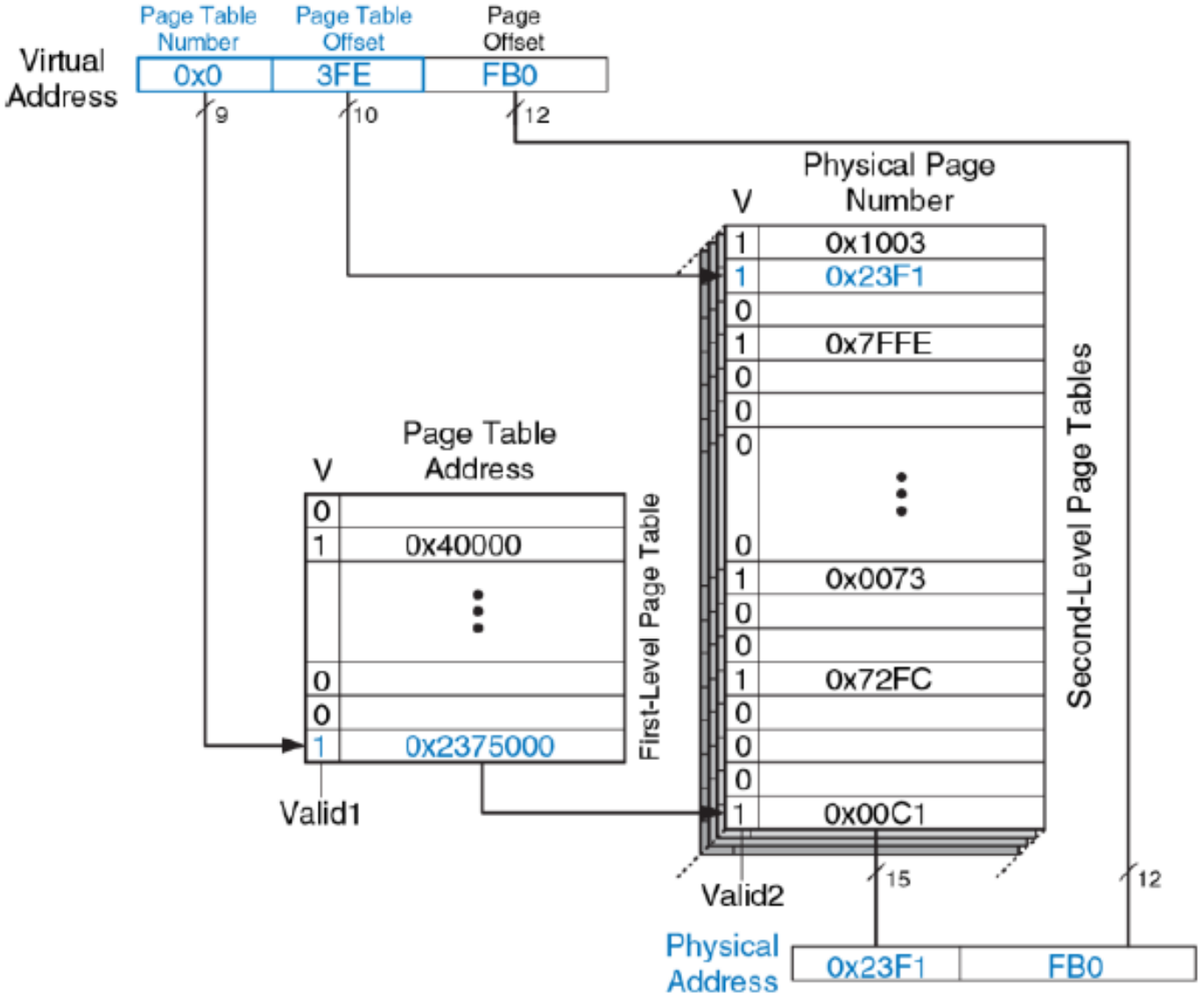
Багаторівневі таблиці сторінок

- Таблиця сторінок першого рівня зберігаються у фізичній пам’яті і вказує де у віртуальній пам’яті зберігаються таблиці сторінок другого рівня
- Малі таблиці сторінок другого рівня зберігаються у віртуальній пам’яті
- Кожна таблиця сторінок другого рівня зберігає інформацію про деякий діапазон віртуальних сторінок



Трансляція адрес з використанням дворівневих таблиць

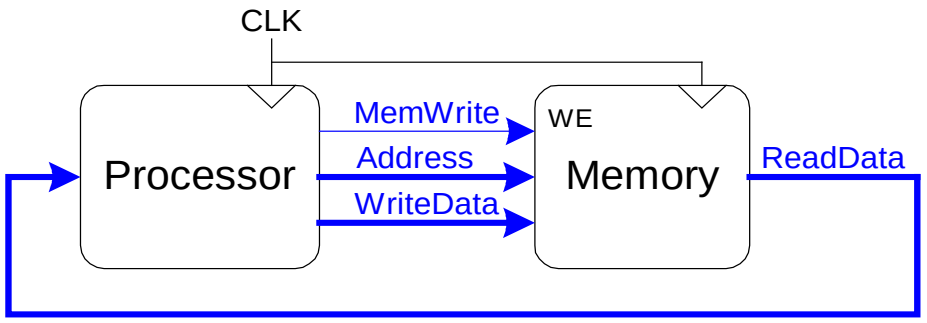
- Віртуальна адреса 0x003FEFB0
- Номер таблиці – старші 9-біт, 0x0 – визначає номер сторінки у таблиці 1-го рівня. Так як V=1 то потрібна таблиця другого рівня знаходиться у пам’яті і її фізична адреса 0x2375000
- Наступні 9-біт віртуальної адреси 0x3FE задають номер рядка у сторінці 2-го рівня . Таблиця 2-го рівня має 1024 (0x400) записів, перенумерованих зверху вниз, тому 1022 (0x3FE) вказує на 2-й рядок. V=1 вказує, що сторінка знаходиться у віртуальній пам’яті, а номер фізичної сторінки 0x23F1.
- Фізична адреса отримується об’єднанням 0x23F1 і зміщення FB0 - 0x23F1FB0



Системи введення-виведення

- Системи введення-виведення підключають комп’ютер до периферійних пристроїв
- Частина адресного простору відводиться під пристрої введення-виведення замість пам’яті, наприклад від 0xFFFF_0000 до 0xFFFF_FFFF.
- Кожному пристрою введення-виведення присвоюється один або декілька адрес в цьому діапазоні. Такий метод зв’язку з пристроями називається введенням-виведенням відображеним у пам’ять (MMIO, memory mapped I/O).
- Дешифратор адреси визначає, який пристрій або пам’ять зв’язується з процесором
- Регістри введення-виведення містять значення, які зчитуються або записуються у периферійні пристрої
- Мультиплексор читання даних: здійснює вибір між пам’яттю або пристроями введення-виведення і встановлює їх як джерело даних, які передаються процесору

Інтерфейс пам’яті



Відображення пристроїв введення-виведення у пам’ять

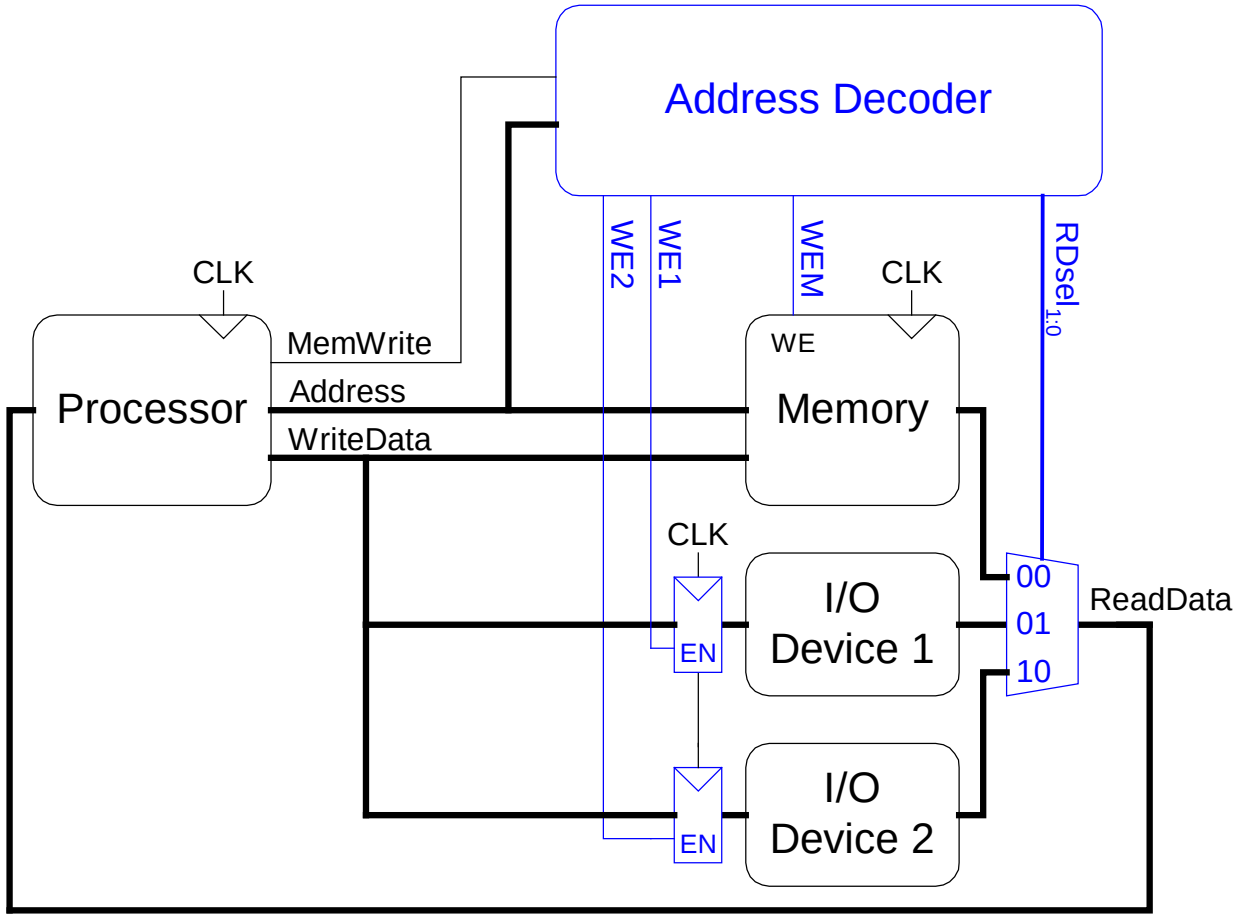
- Пристрою введення-виведення Device 1 призначена адреса пам’яті 0xFFF_FFF4
- Код асемблера для MIPS для запису значення 7 в Device 1 і читання даних з Device 1

```
addi $t0, $0, 7
#FFF4 розширюється знаком до 0xFFFFFFF4
sw $t0, 0xFFF4($0)
```

- Дешифратор адреси встановлює WE1, так як адреса 0xFFFF_FFF4 і MemWrite є TRUE. призначена адресі пам’яті 0xFFF_FFF4. Дані (значення 7) по шині WriteData записуються в регістр, який підключений до входу Device 1.
- Читання даних з пристрою Device 1

```
lw $t1, 0xFFF4($0)
```

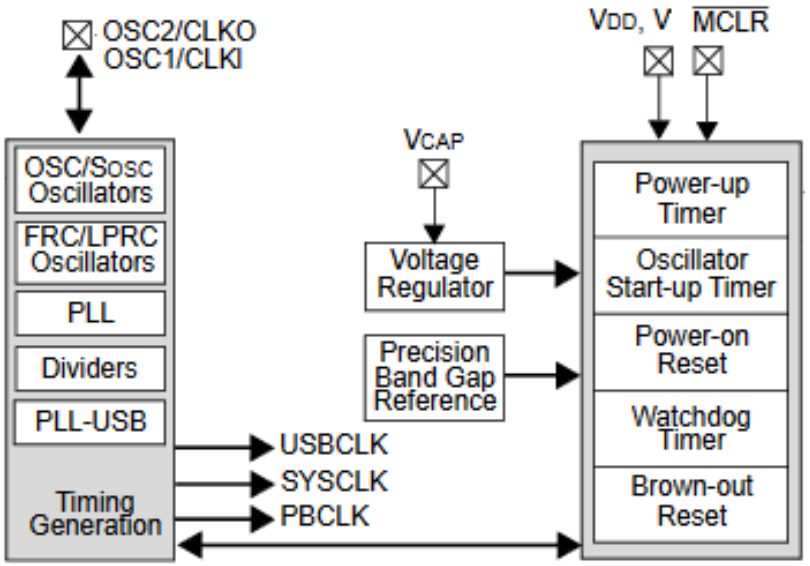
- Дешифратор адреси встановлює Rdsel 1:0 в 01, оскільки він визначає, що адреса 0xFFFF_FFF4 і значення MemWrite – False. Вихідні дані із пристрою проходять через мультиплексор на шину ReadData і завантажуються в регістр $t1 процесора

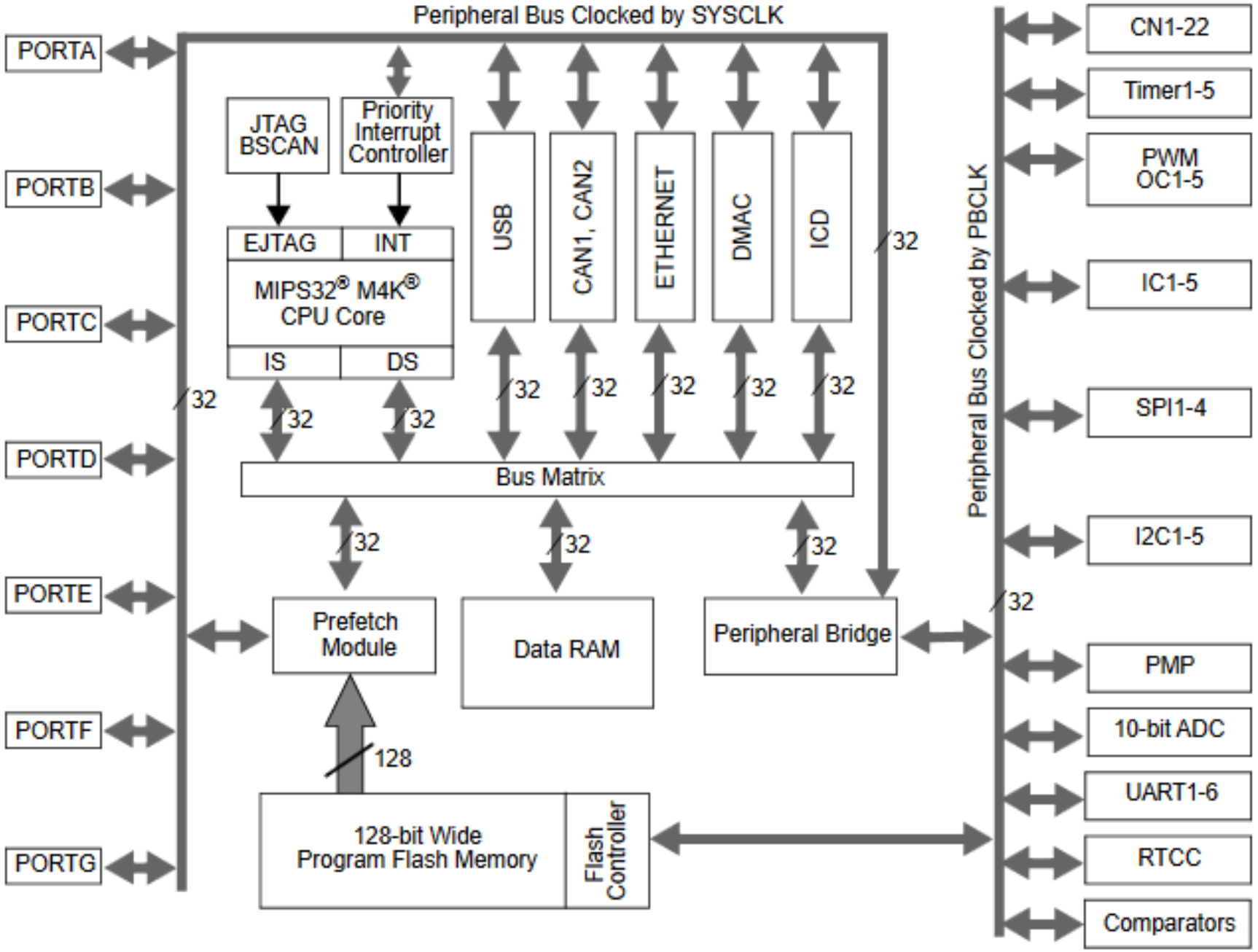


Вбудовані підсистеми введення-виведення

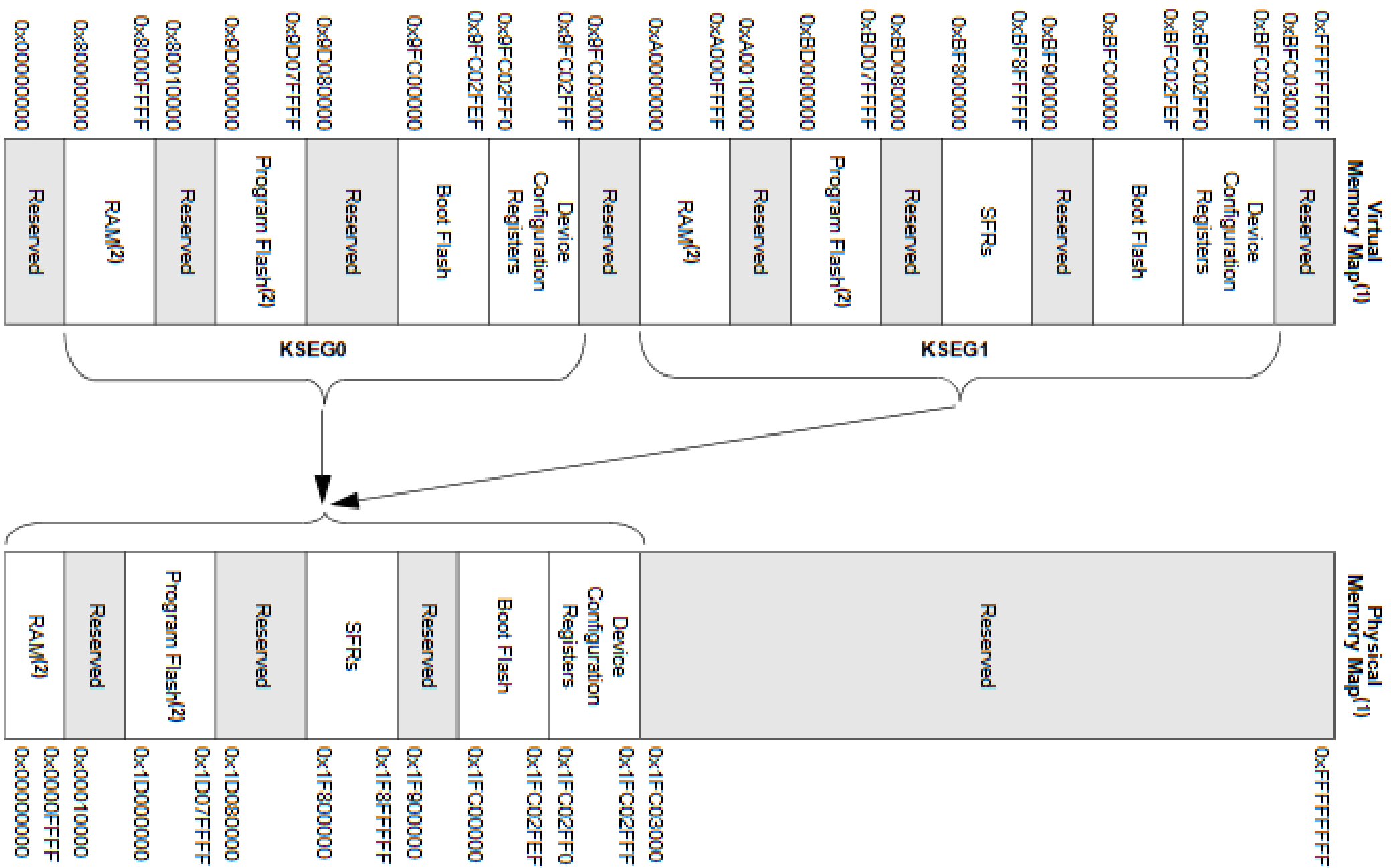
- Вбудовані системи використовують процесор для керування взаємодією з фізичним середовищем. Звичайно вони будуються на основі мікроконтролерів, які поєднують мікропроцесор з набором простих у використанні периферійних пристроїв:
  - цифрові і аналогові виводи
  - послідовні порти
  - таймери
  - аналого-цифрові перетворювачі
- За приклад можна взяти мікроконтролер PIC32MX675F512H, який використовує 32-розрядний MIPS процесор.

Блок схема мікроконтролера





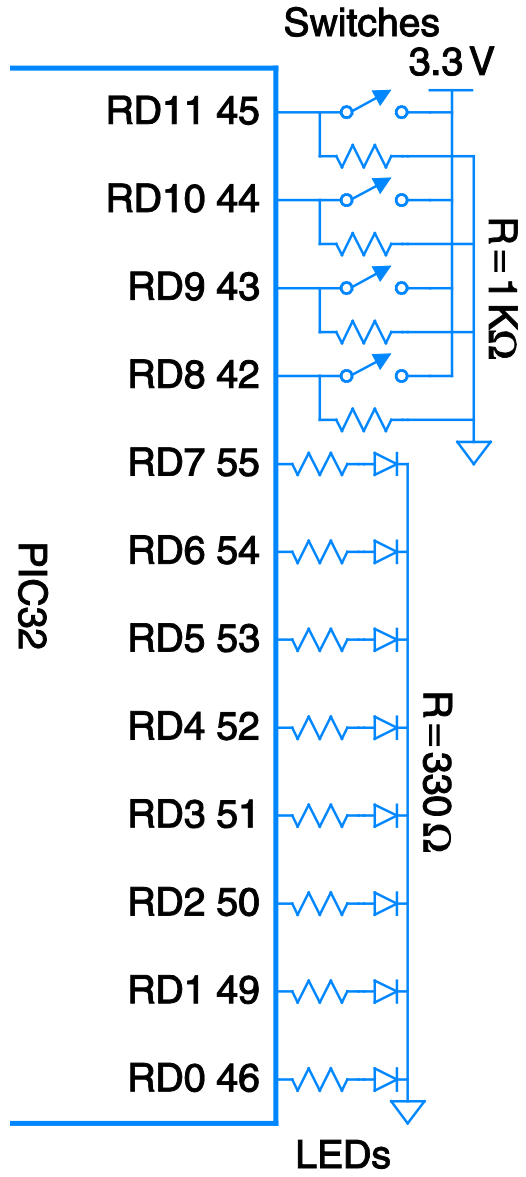
Карта пам’яті мікроконтролера



Порти введення-виведення загального призначення (GPIO

- GPIO порти використовуються для читання-запису цифрових сигналів. Кожний порт може мати до 16 виводів.
- Приклад підключення світлодіодів і перемикачів до порту D.
- Приклад програми С, яка читає стан чотирьох перемикачів і вмикає чотири нижніх світлодіоди

```
// C код
#include <p3xxxx.h>
int main(void) {
  int switches;
  TRISD = 0xFF00;      // RD[7:0] outputs
                       // RD[11:8] inputs
  while (1) {
    // read & mask switches, RD[11:8]
    switches = (PORTD >> 8) & 0xF;
    PORTD = switches;  // display on LEDs
  }
}
```

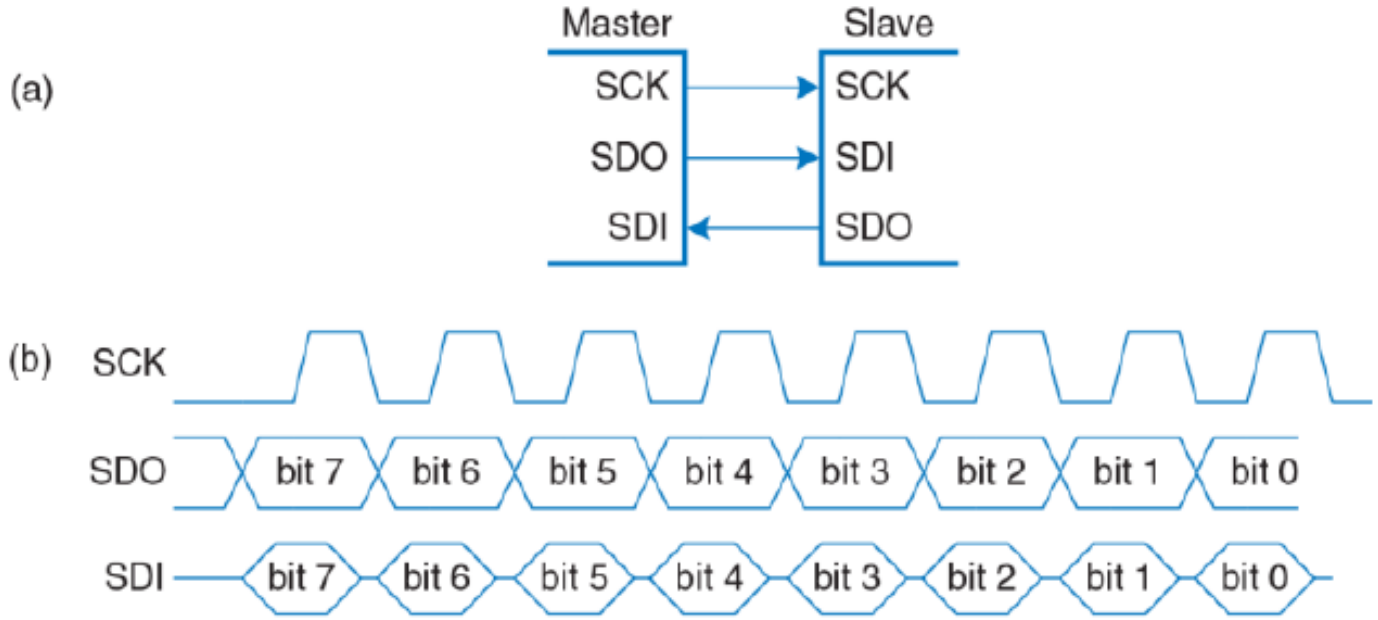


Послідовне введення-виведення

- Послідовне введення-виведення використовує менше число ліній передачі
- Приклад послідовних протоколів:
  - послідовний периферійний інтерфейс (англ. Serial Peripheral Interface, **SPI**)
  - універсальний асинхронний приймач-передавач (англ. Universal Asynchronous Receiver/Transmitter, **UART**)
  - двопровідна двонаправлена шина, (**I^2C**)
  - Універсальна послідовна шина, (**USB**)
  - **Ethernet**

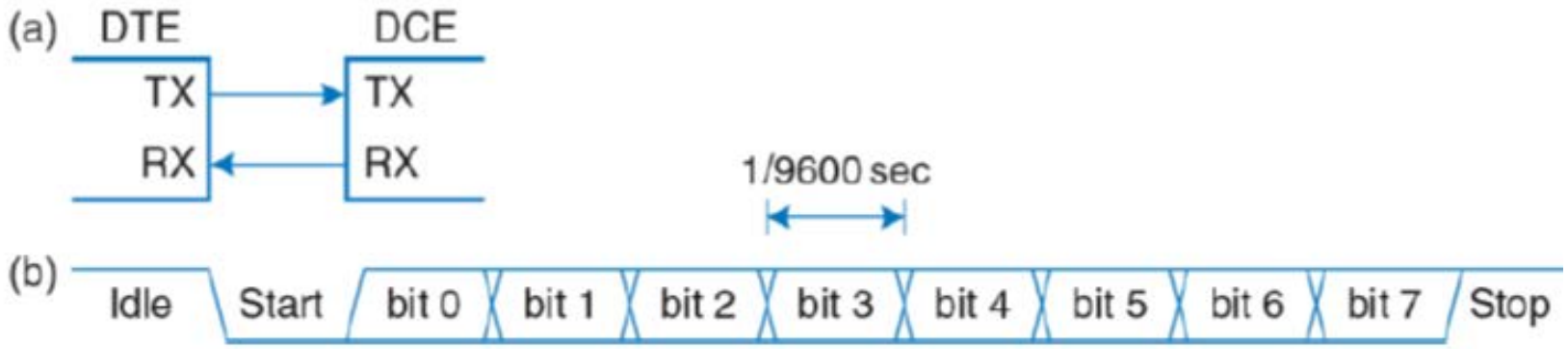
Послідовний периферійний інтерфейс SPI

- Послідовний периферійний інтерфейс SPI використовує тактові імпульси SCK
- Ведучий пристрій (master) ініціює встановлення зв’язку з веденим пристроєм (slave) шляхом генерації імпульсів на вивід SCK
- Ведучий пристрій посилає дані на вивід SDO (Serial Data Out – послідовний вихід даних) веденому пристрою, починаючи із старшого біту
- Ведений пристрій може послати дані (на вивід SDI) ведучому пристою, починаючи із старшого біту



Універсальний приймач-передавач UART

- UART є периферійним пристроєм послідовного введення-виведення без використання тактового імпульсу
- Передавач і приймач наперед домовляються про швидкість передачі даних
- Швидкість передачі даних: 300, 1200, 2400, 9600, …115200 бод
- Звичайні параметрм: 1 стартовий біт (0), 8 бітів даних, без контролю парності, 1 стоповий біт. Всього 10 біт, що відповідає швидкості передачі 9600 бод => 9600 символів/сек => 960 знаків/сек => 960×8=7680 біт/сек.
- Розширені параметри: 1 стартовый бит (0), 7-8 бітів данных, біт парності (опційний), 1 і більше стопових бітів
- Линія простоює при високому логічному рівні



Таймери

- Для вимірювання часу у вбудованих системах використовують таймери. PIC32 має п’ять 16-розрядних таймери.
- Кожний таймер працює як 16-розрядний лічильник, який накопичує такти внутрішнього периферійного годинника (20 МГц у даному випадку).
- Таймер у режимі закритого накопичення часу веде відлік, поки на зовнішньому виводі підтримується високий рівень. Це дозволяє вимірювати тривалість зовнішнього імпульсу.
- Приклад С-функції, яка створює затримку в 1 мілісекунду:

```
#include <P32xxxx.h>
void delaymicros(int micros) {
  if (micros > 1000) {          // avoid timer overflow
    delaymicros(1000);
    delaymicros(micros-1000);
  }
  else if (micros > 6){
    TMR1 = 0;                   // reset timer to 0
    T1CONbits.ON = 1;           // turn timer on
    PR1 = (micros-6)*20;        // 20 clocks per microsecond
                                // Function has overhead of ~6 us
    IFS0bits.T1IF = 0;          // clear overflow flag
    while (!IFS0bits.T1IF);     // wait until overflow flag set
  }
}
void delaymillis(int millis) {
  while (millis--) delaymicros(1000); // repeatedly delay 1 ms
```

Аналогове введення-виведення

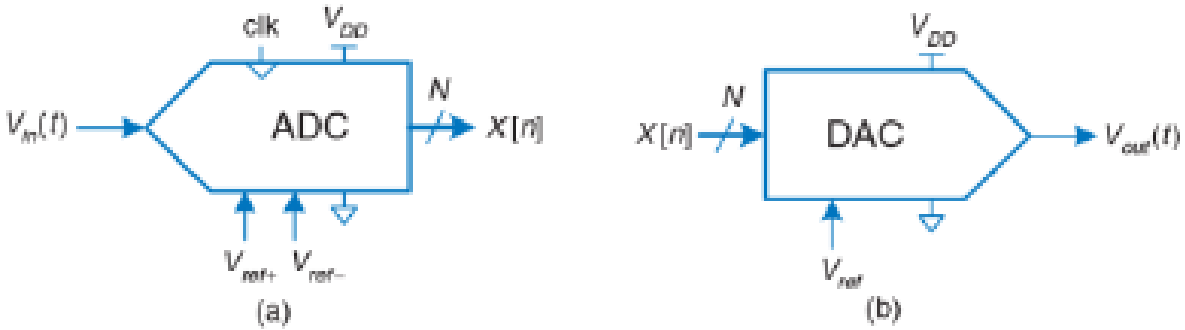
Вбудовані системи для взаємодії з фізичним середовищем використовують: аналогове введення-виведення:

- Аналогове введення: аналого-цифрові перетворювачі (АЦП) квантують аналогові сигнали у цифрові значення. Вхідний сигнал в межах V_{ref-}-V_{ref+} перетворюється до 0-2^{N-1} . Залежність між напругою аналогового входу *Vin(t)* і цифровою вибіркою *X[n]*

$$X[n]=2^N \frac{V_{in}(t)-V_{ref-}}{V_{ref+}-V_{ref-}}, \text{ де } n=\frac{t}{f_s}, f_s \text{ - частота дискретизації}$$

- Аналогове виведення:
  - цифро-аналогові перетворювачі (ЦАП) перетворюють цифрові значення в аналоговий сигнал. Цифрове значення в межах 0-2^{N-1} перетворюється до V_{ref-}-V_{ref+}

    Вихідна напруга $V_{out}(t) = \frac{X[n]}{2^N} V_{ref}$
  - широтна-імпульсна модуляція
- Умовні позначення АЦП (a) і ЦАП (b):



Широтна-імпульсна модуляція

Широтна-імпульсна модуляція (ШІМ, англ. Pulse width modulation, PWM) дозволяє генерувати аналоговий сигнал у цифровій системі за рахунок зміни частини періоду з високим рівнем. Відношення частини періоду з високим рівнем до тривалості періоду називається коефіцієнтом заповнення (duty cycle). Середнє значення напруги на виході пропорційне коефіцієнту заповнення. Низькочастотна фільтрація сигналу усовує коливання і вихідний сигнал приймає потрібне середнє значення. Частота зрізу фільтру низьких частот:

$$f_c = \frac{1}{2\pi RC}$$