POLITECNICO DI MILANO





Introduzione all'hardware

Laboratorio Software 2008-2009
C. Brandolese

L'hardware

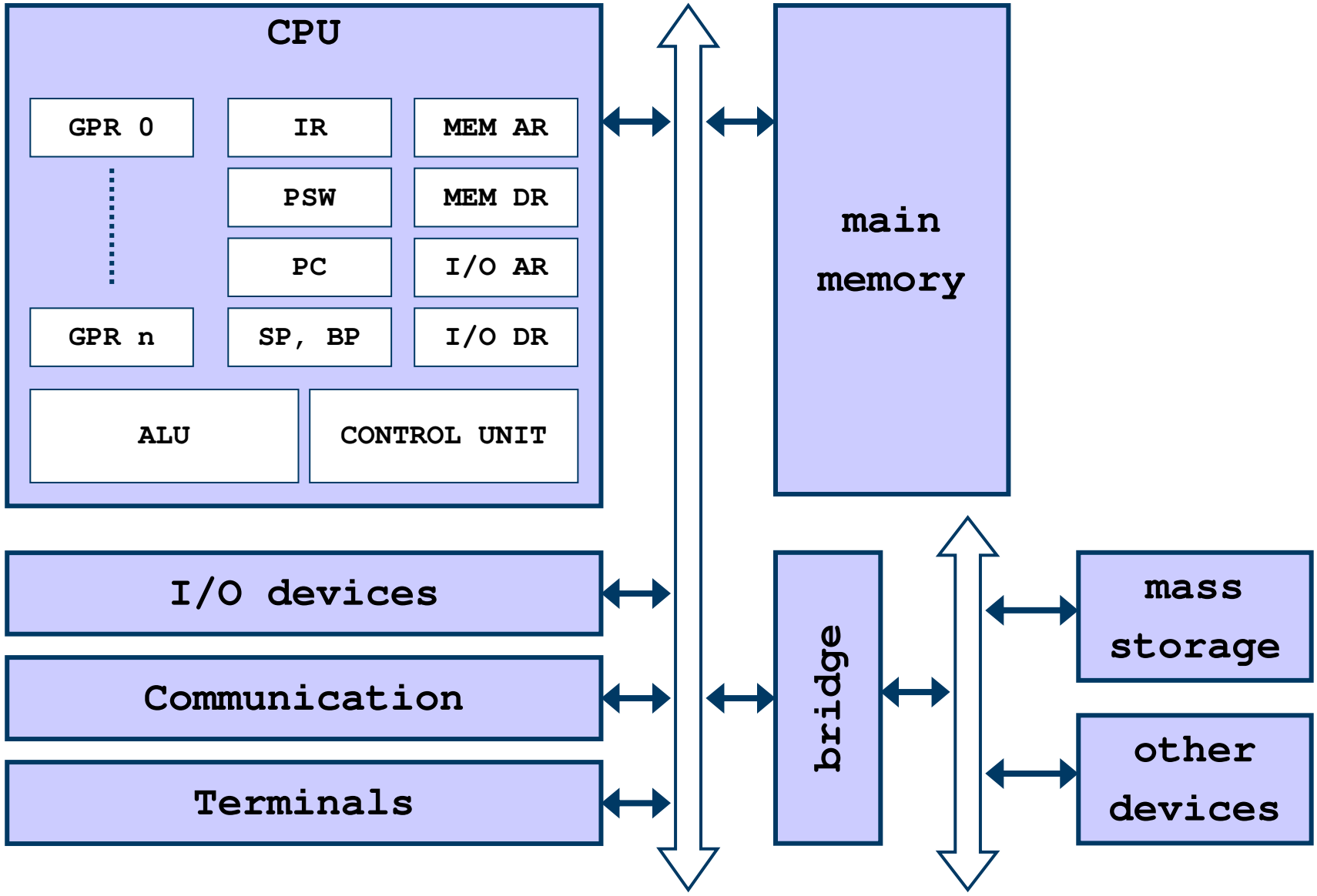
Costituisce il supporto all'esecuzione del software

- Attraverso i servizi del sistema operativo

Si compone di diversi elementi fondamentali

- Processore
  - Esecutore del codice
- Memoria principale
  - Di grandi dimensioni
  - Volatile
- Moduli di I/O
  - Dispositivi per la comunicazione
  - Memorie secondarie
  - Terminali
- Bus

Struttura di un calcolatore



Registri

Registri speciali

- Non tutti accessibili al programmatore
- Utilizzati a livello hardware per realizzare
  - Ciclo di esecuzione
  - Accesso alla memoria
  - Accesso ai dispositivi
- Utilizzati dal sistema operativo
  - Per realizzare il controllo dell'esecuzione dei programmi

Registri generici

- Accessibili al programmatore
  - Il loro uso è ottimizzato dal compilatore
- Utilizzati dai programmi per ridurre gli accessi alla memoria
  - Contengono dati temporanei

Regsitri indirizzo

Base pointer (BP)

- Indirizzo della base di una area di memoria locale

Segment pointer

- Nei sistemi con memoria segmentata punta alla base di un segmento

Stack pointer (SP)

- Punta al top dello stack

Memory address register (MAR)

- Indirizzo per l'accesso alla memoria principale

I/O address register (IOAR)

- Indirizzo per l'acesso ai dispositivi di I/O

Program counter (PC)

- Indirizzo della prossima istruzione nel flusso di esecuzione

Registri dati

Instruction register

- Contiene l'istruzione correntemente in esecuzione

Memory data register (MDR)

- Registro dati per l'accesso alla memoria

I/O data register (IODR)

- Registro dati per l'accesso ai dispositivi di I/O

Program status word (PSW)

- Condition codes
  - Positive, negative, zero, overflow, carry, ...
- Interrupt
  - Enarble, disable
- Mode
  - User, supervisor

Ciclo istruzione

Il processore legge la prossima istruzione (Fetch)

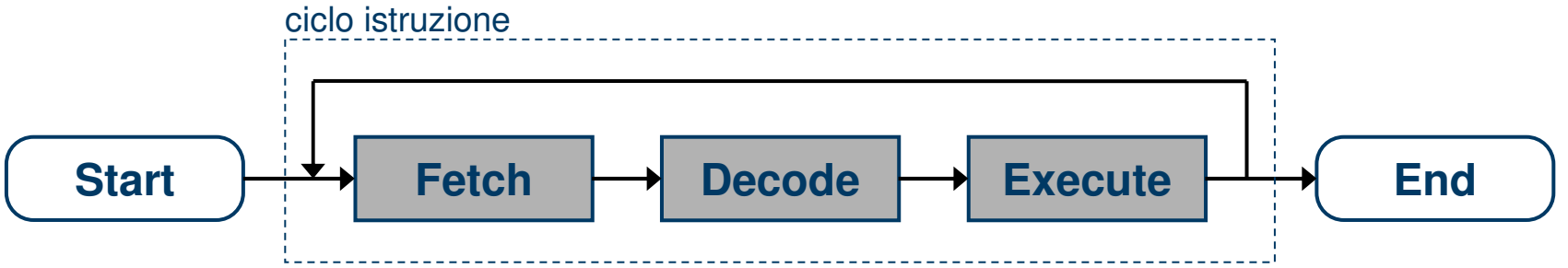
- L'indirizzo dell'istruzione è contenuta nel program counter
- Il program counter viene incrementato dopo la lettura
- L'istruzione è memorizzata nell'instruction register

L'unità di controllo decodifica l'istruzione (Decode)

- Genera i segnali di controllo necessari

Il processore esegue l'istruzione (Execute)

- Utilizza la ALU e i registri generici



Tipi di istruzioni

Memoria-Processore

- Trasferiscono dati tra processore e memoria
- Utilizzano la coppia di registri MAR/MDR

Memoria-Dispositivi I/O

- Trasferiscono dati tra processore e dispositivi di I/O
- Utilizzano la coppia di registri IOAR/IODR

Elaborazione

- Eseguono istruzioni artimetiche o logiche
- Utilizzano la ALU e i registri generici

Controllo

- Modificano il flusso di esecuzione
- Utilizzano i registri PC, SP

Interrupt

Supponiamo che una periferica che riceve dati in modo irregolare debba essere servita dal processore

- Polling
  - Il processore interroga la periferica periodicamente per verificare se sono disponibili nuovi dati
  - Grande spreco di tempo
- Interrupt
  - La periferica segnala al processore la disponibilià di dati
  - Richiede uno o più pin aggiuntivi sul processore (INT1, INT2, ...)

Quando si verifica un interrupt

- Il processore sospende l'esecuzione del programma corrente
- Salta ad una opportuna Interrupt Service Routine (ISR)
- Svolge l'operazione di I/O richiesta
- Riprende l'esecuzione del programma interrotto

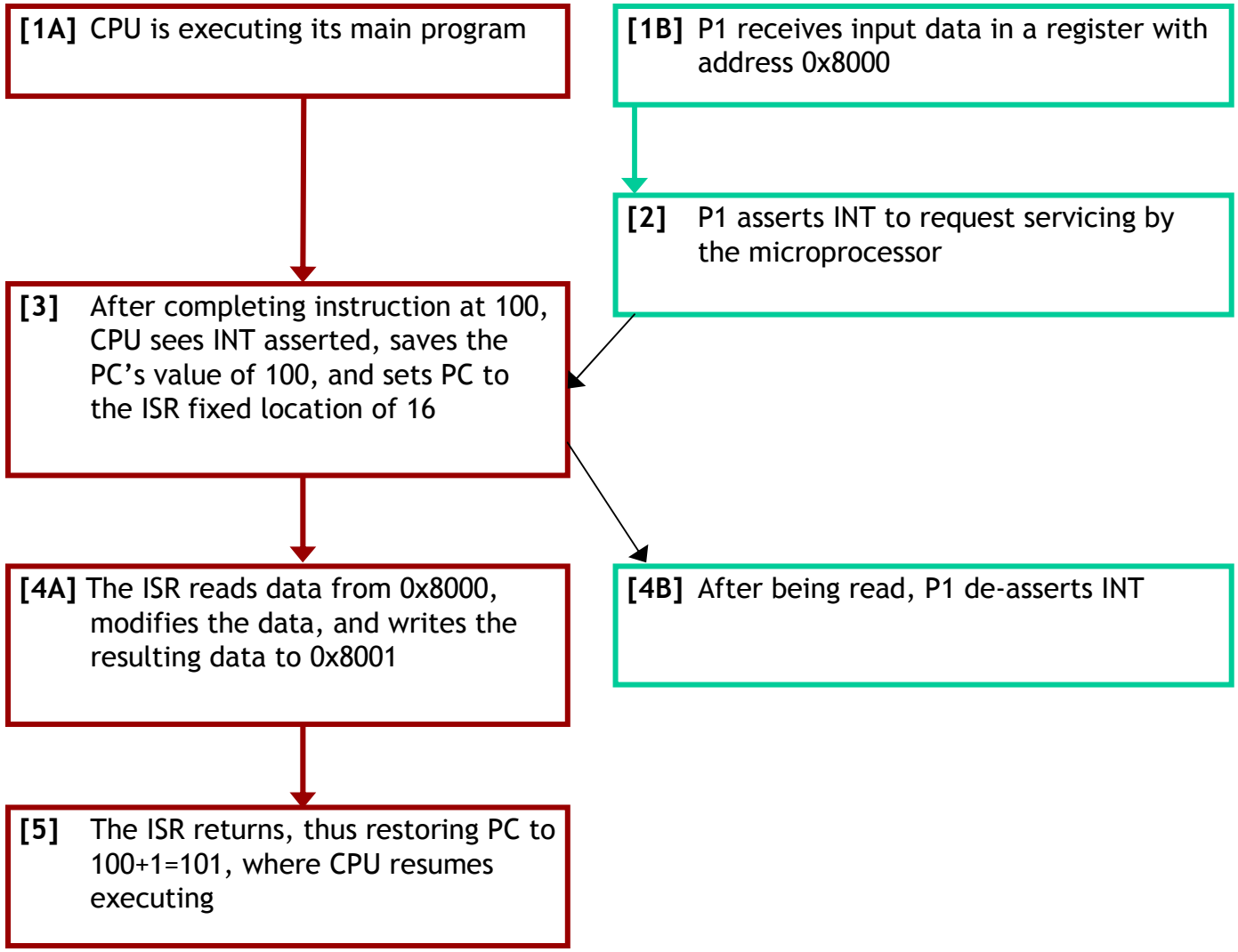
Interrupt

Differenti seganli di interrupt (INT0, INT1, …) hanno differenti interrupt service routines (ISR0, ISR1, …)

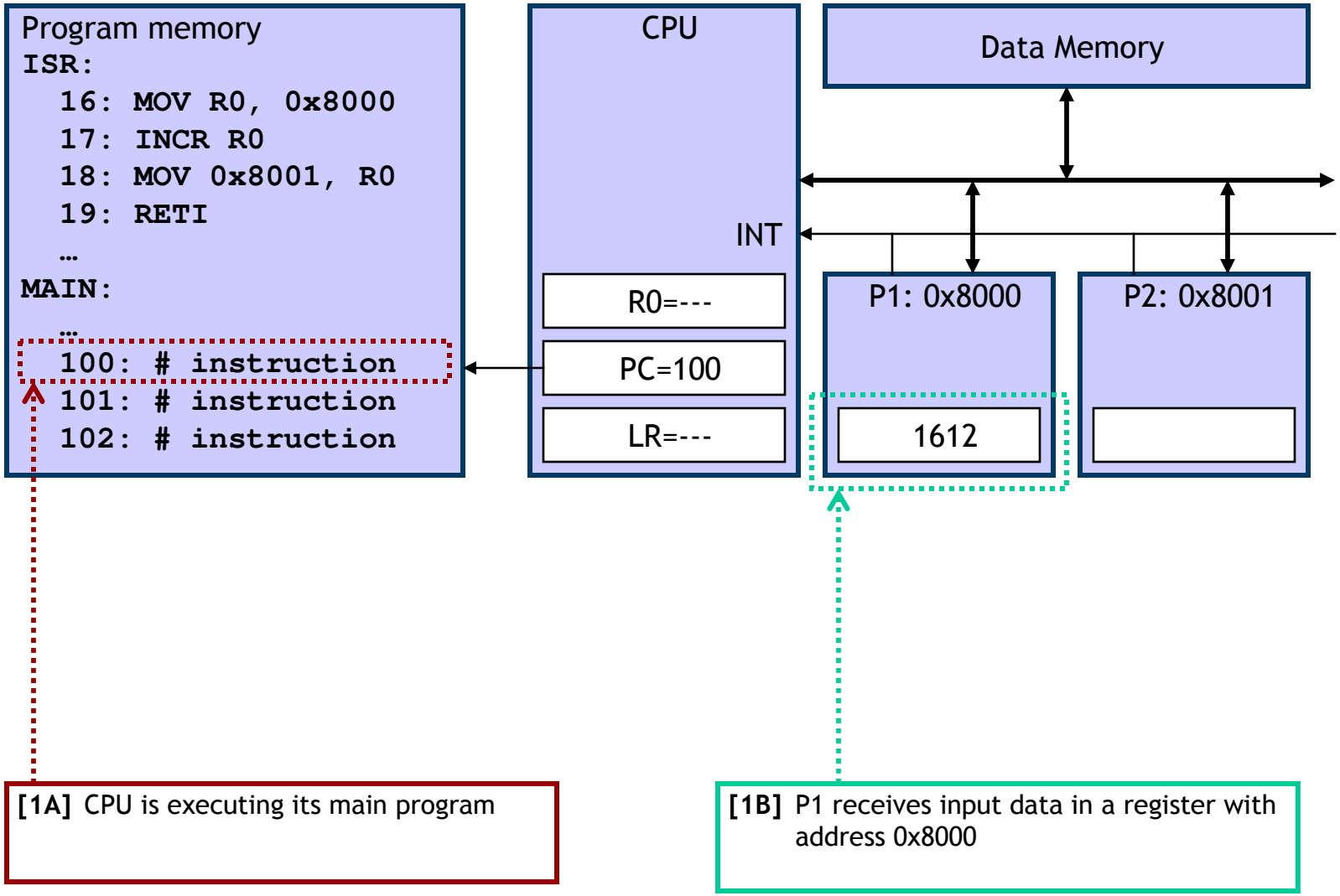
Come si determina la ISR corretta per un dato interrupt?

- Fixed interrupt
  - L'indirizzo dell'ISR è cablato nel processore e non può cambiare
  - A tale indirizzo si può trovare la routine vera e propria o una istruzione di salto ad una routine memorizzata altrove
- Vectored interrupt
  - Le periferiche devono fornire l'indirizzo della ISR corrispondente
  - Di uso comune quando vi sono diverse periferiche connesse al bus
- Compromesso: Interrupt address table
  - La tabella si trova in una regione fissa della memoria
  - I dispositivi forniscono un indice della tabella invece di un indirizzo

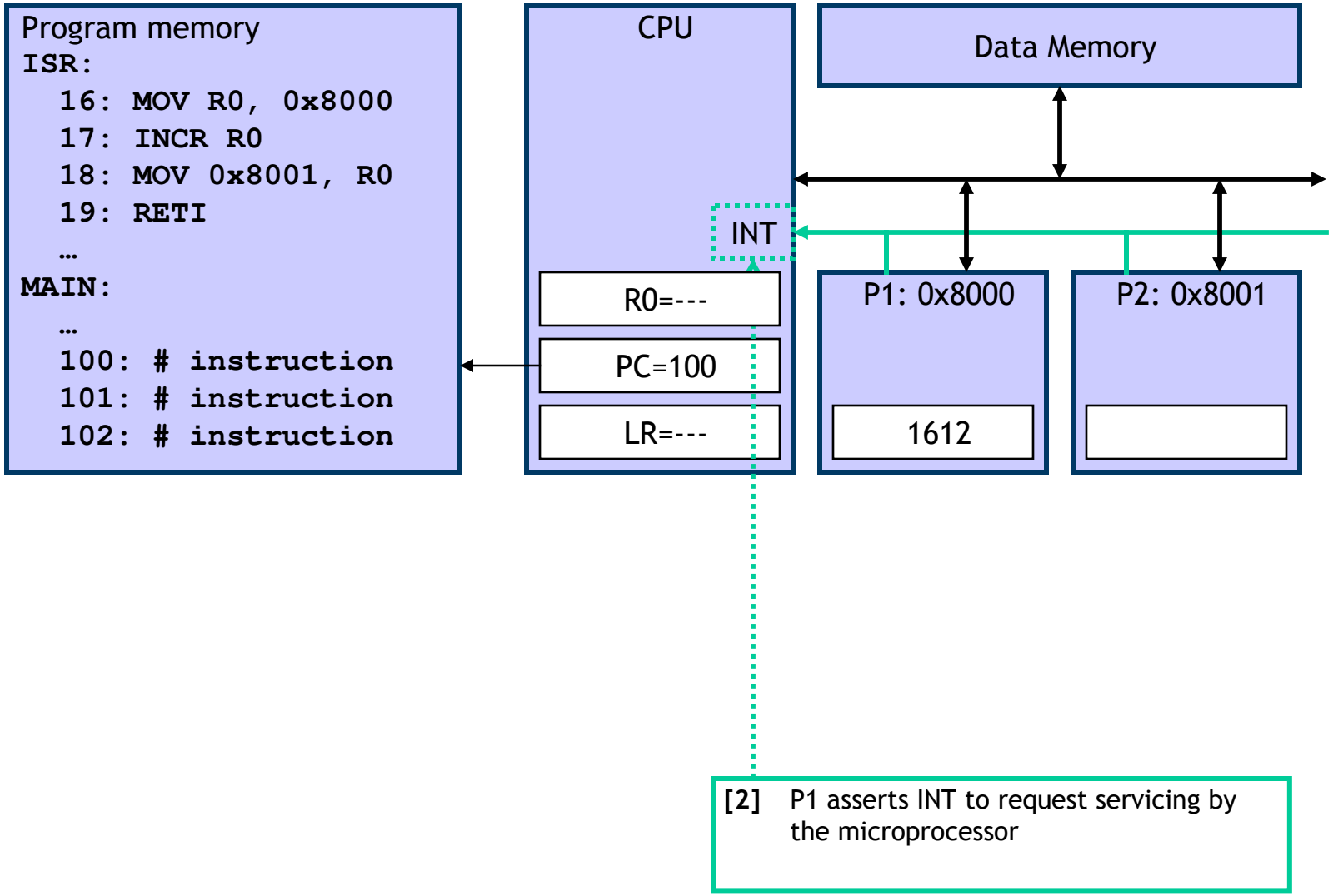
Fixed interrupt



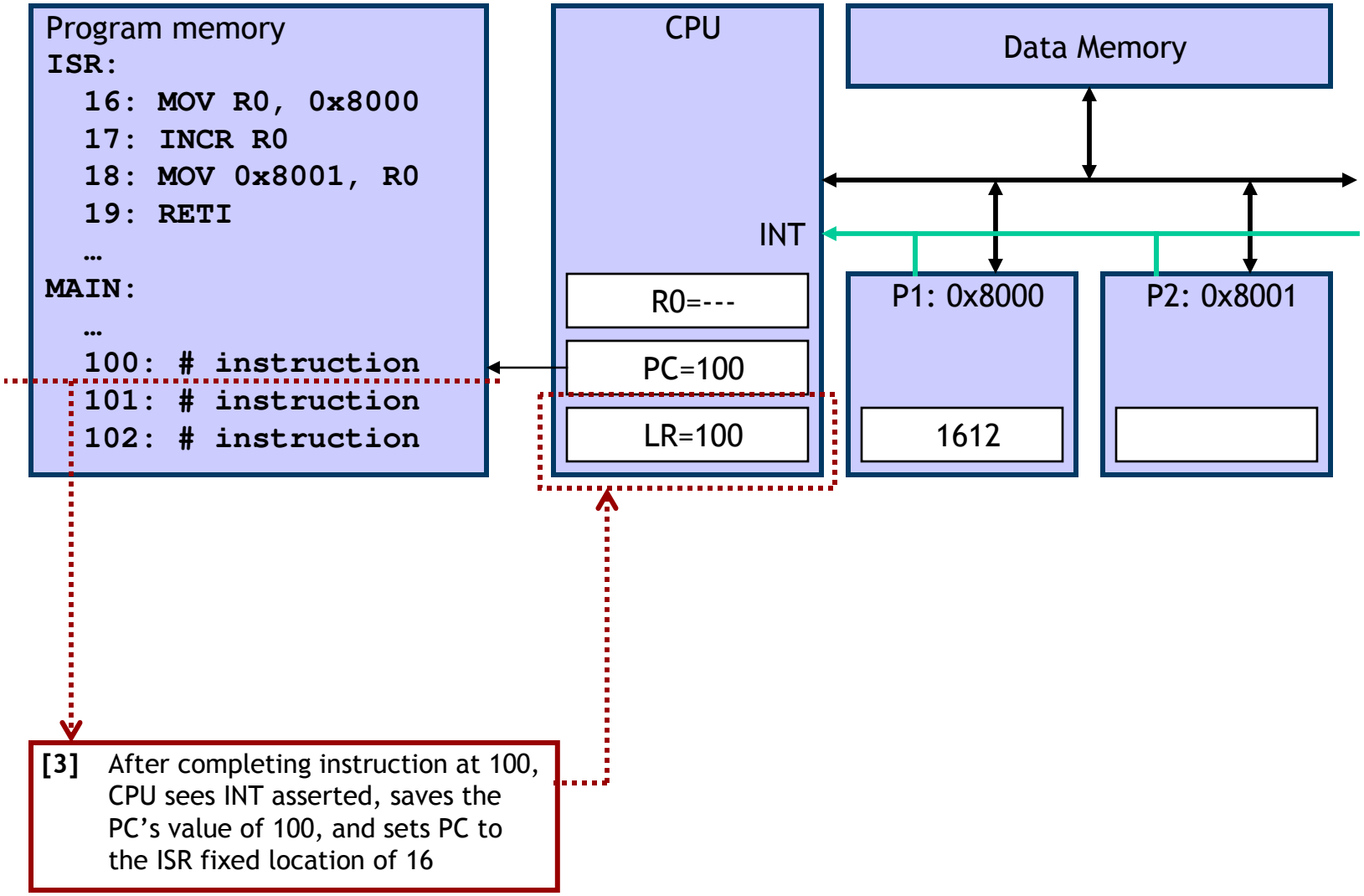
Fixed interrupt



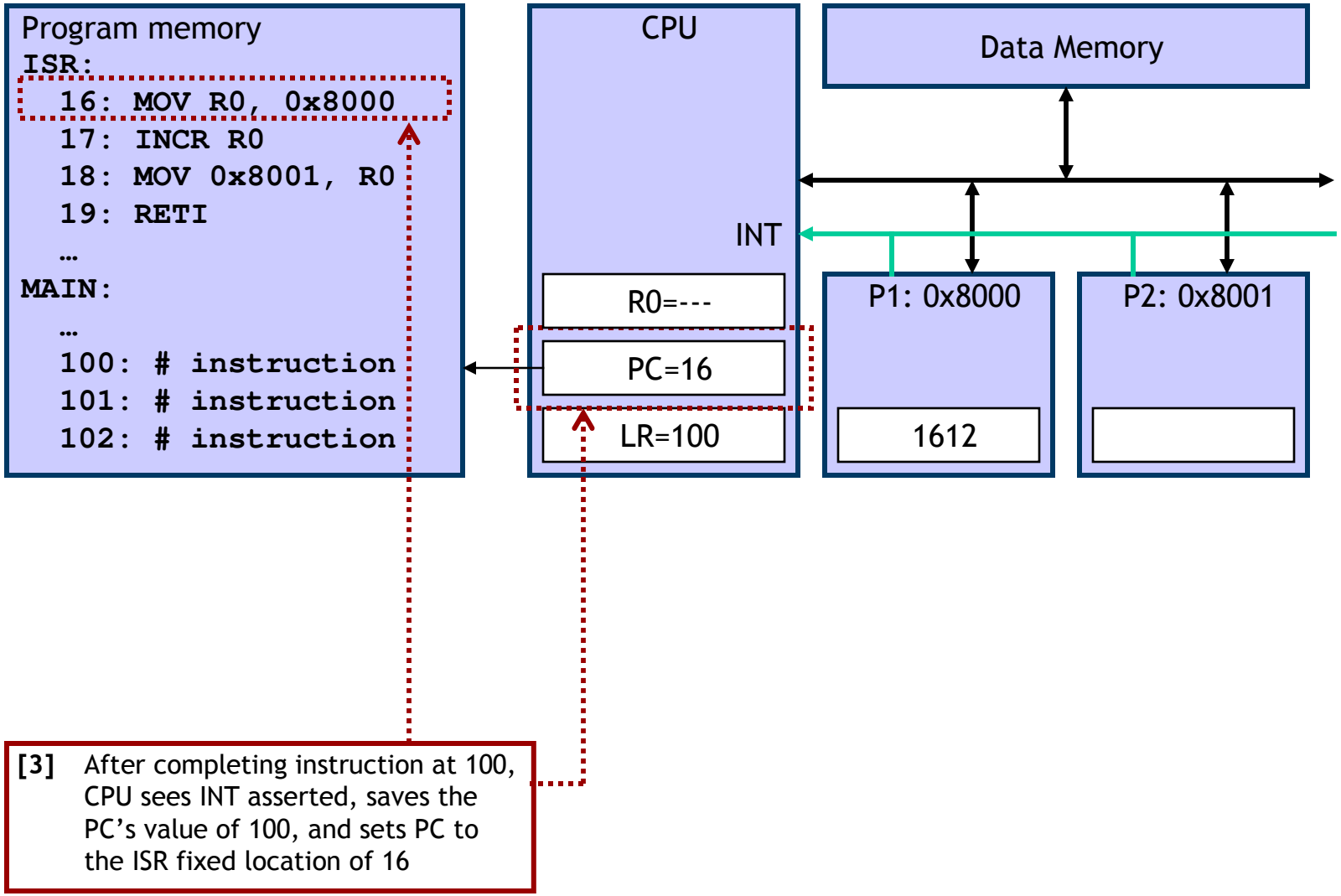
Fixed interrupt



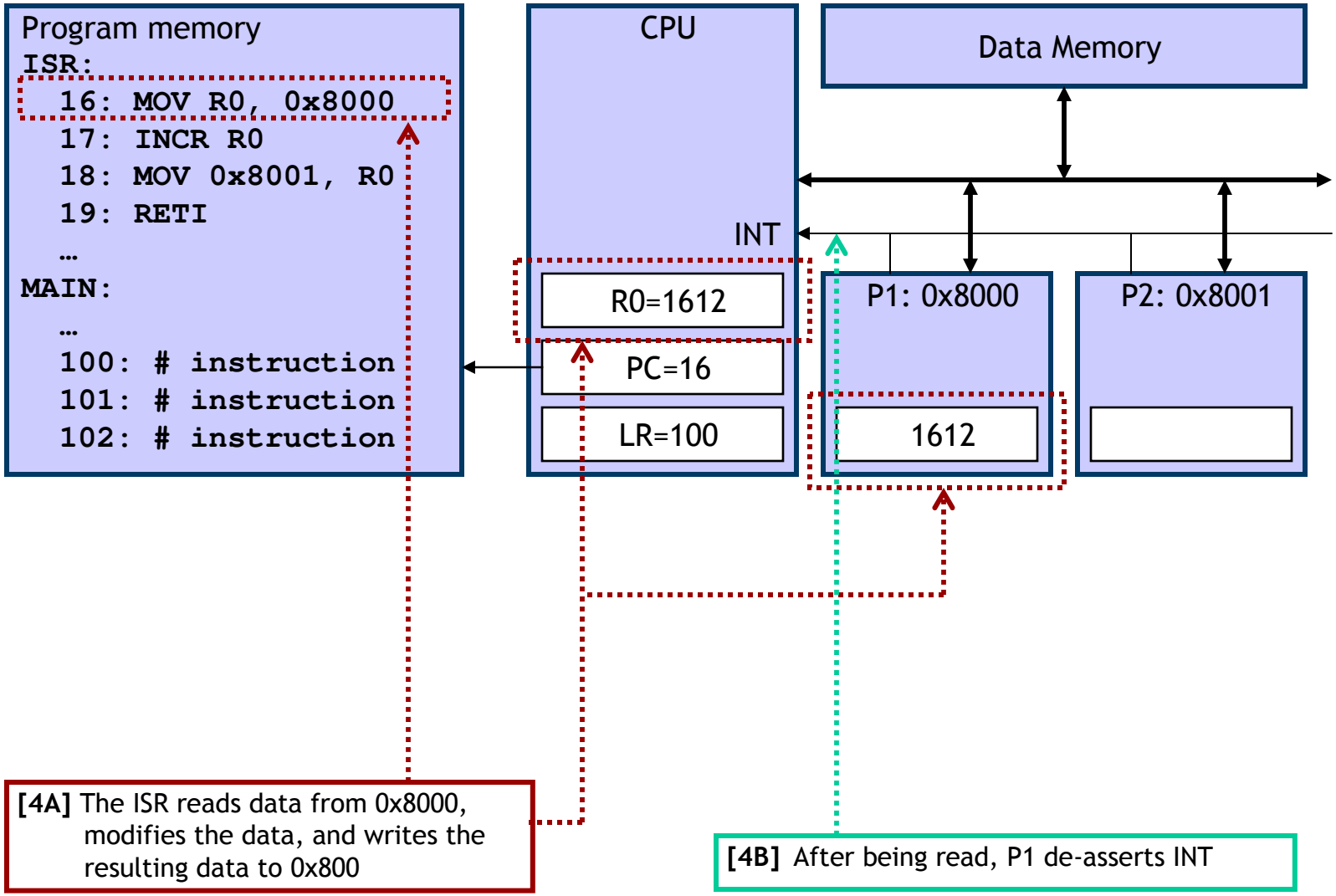
Fixed interrupt



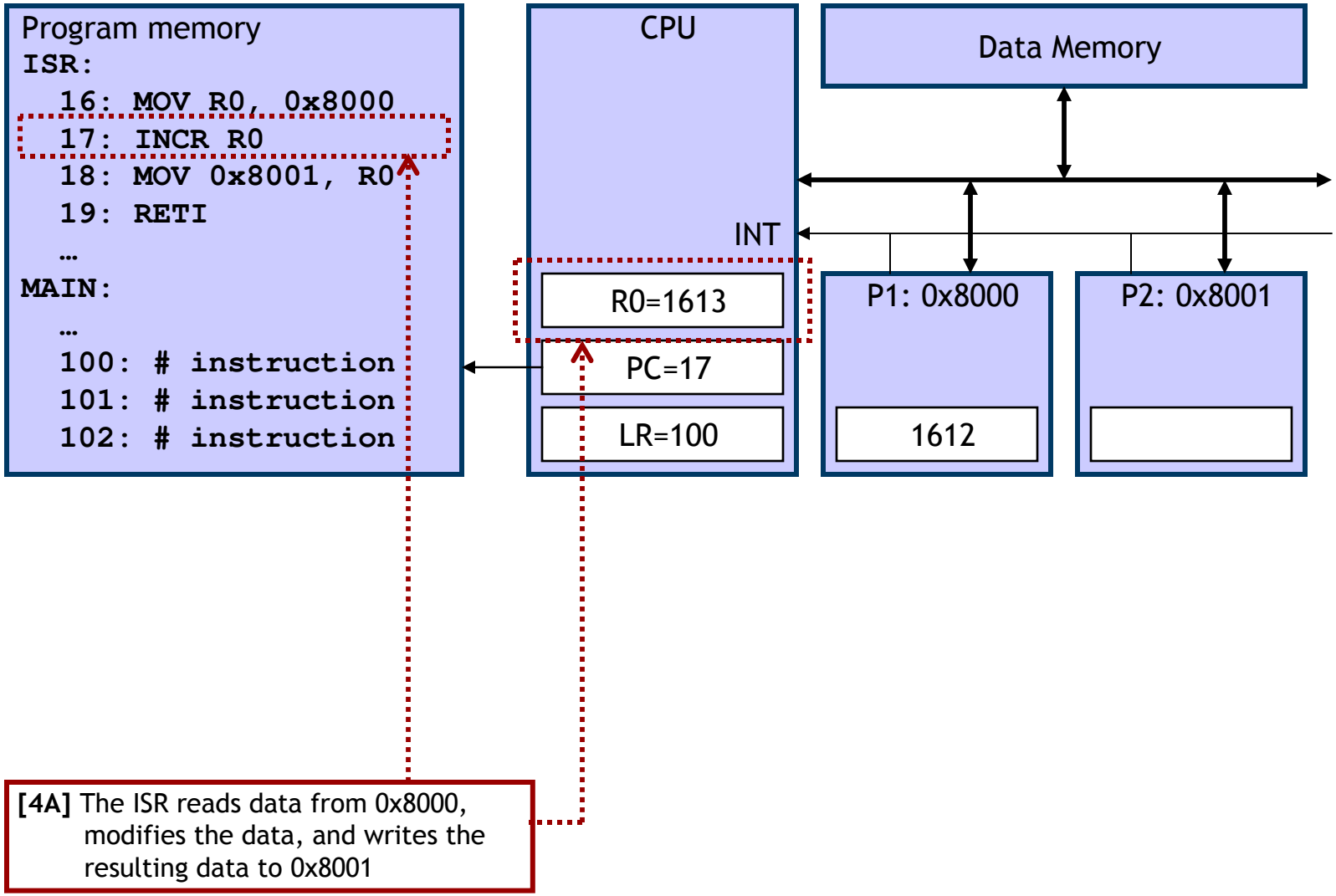
Fixed interrupt



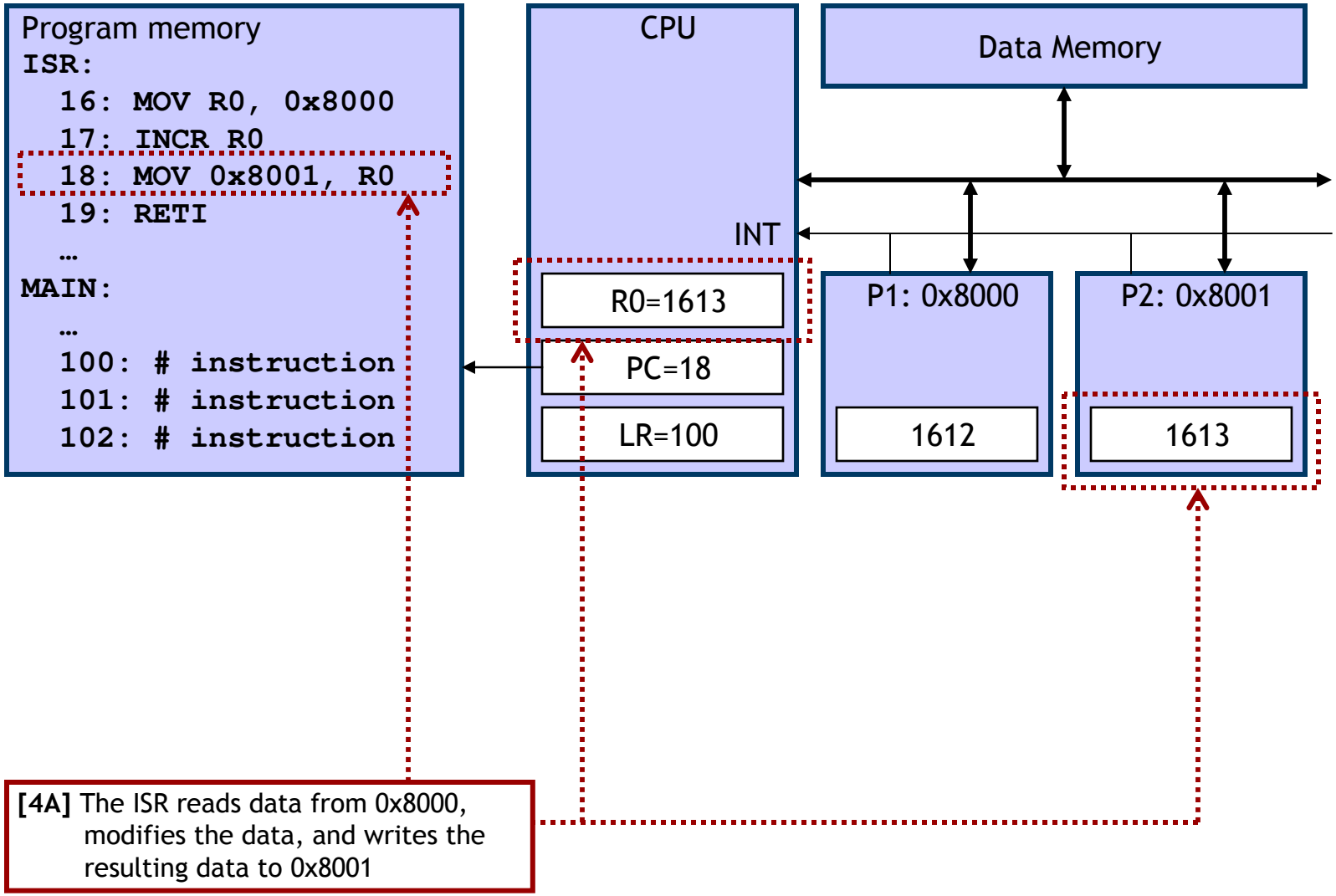
Fixed interrupt



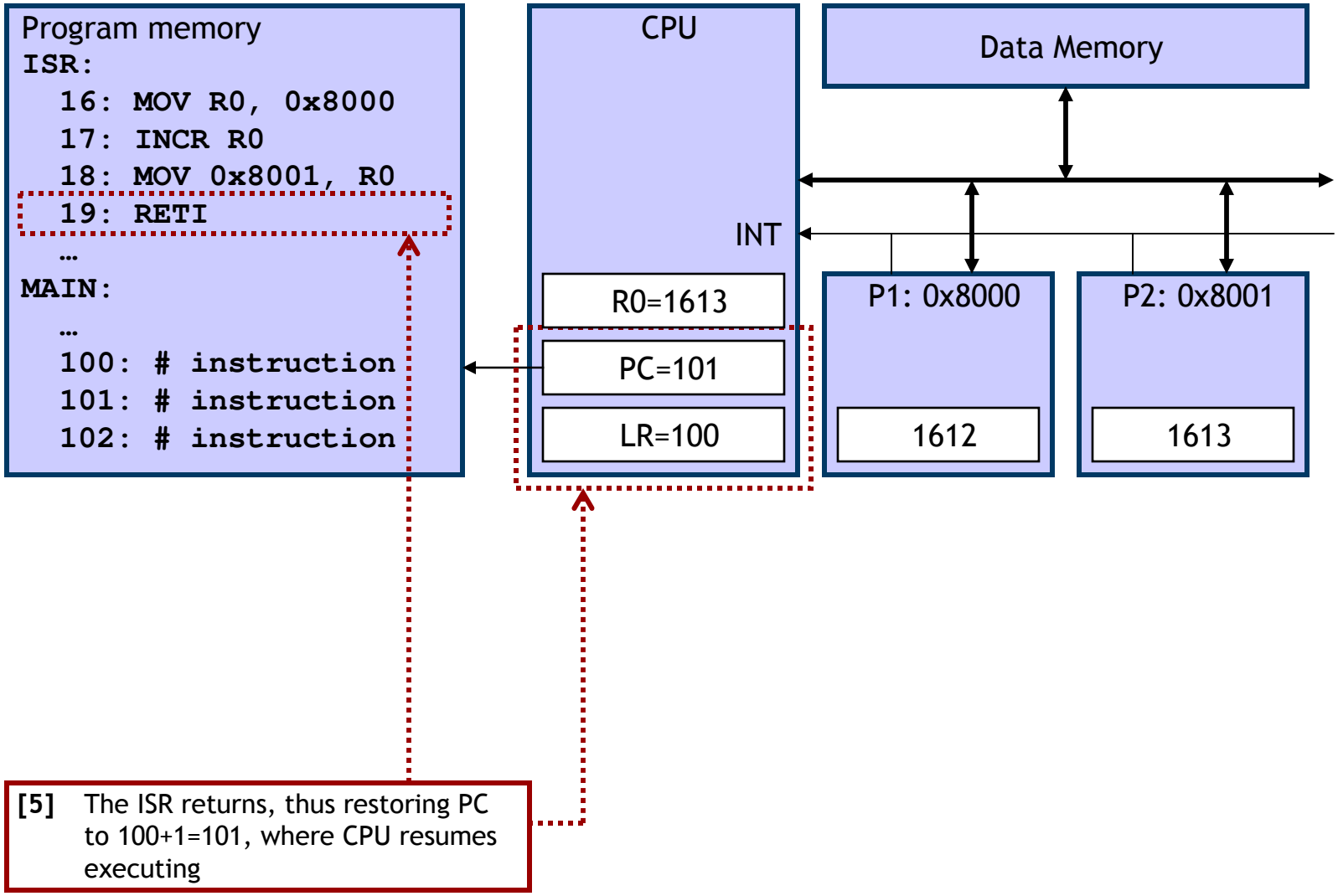
Fixed interrupt



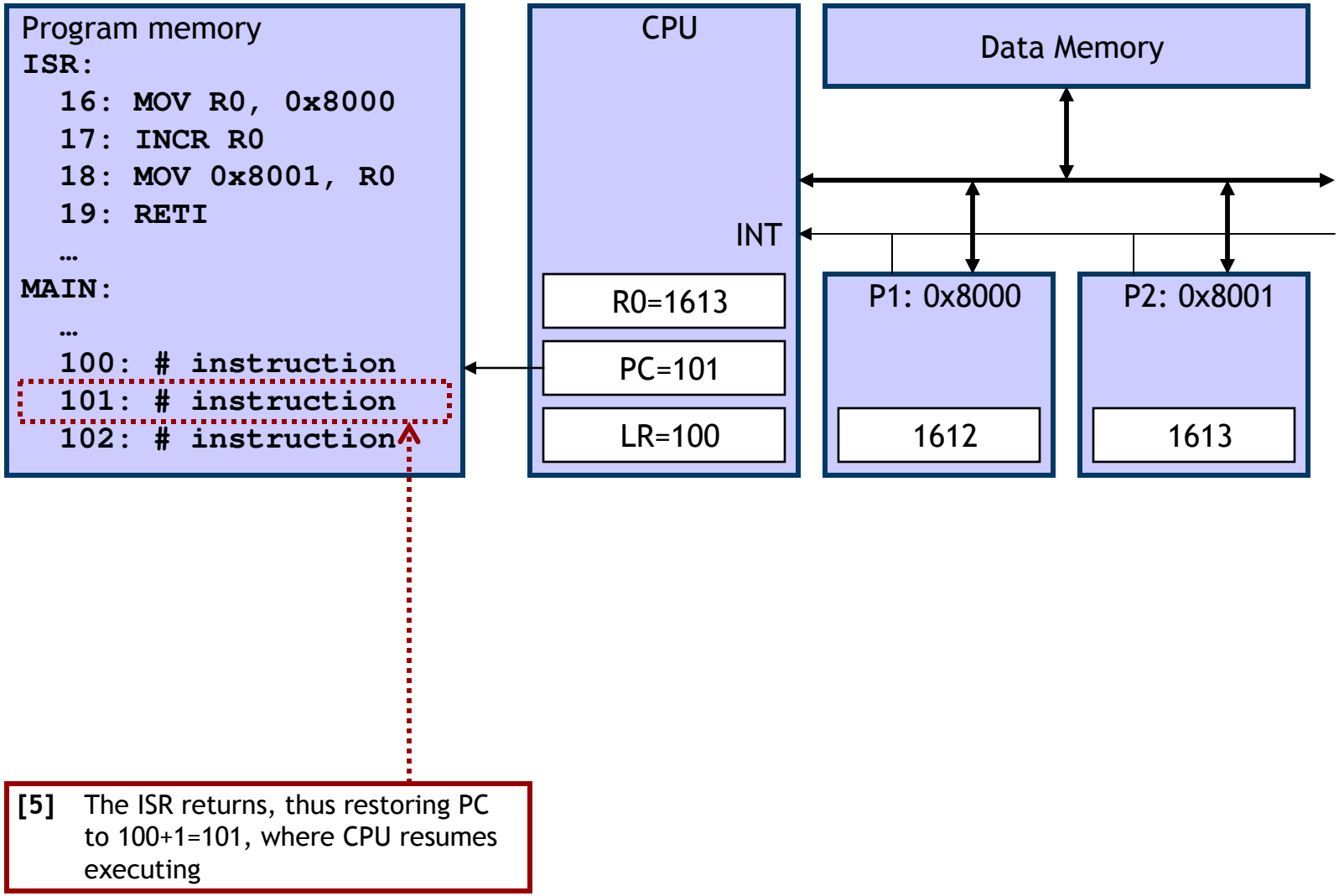
Fixed interrupt



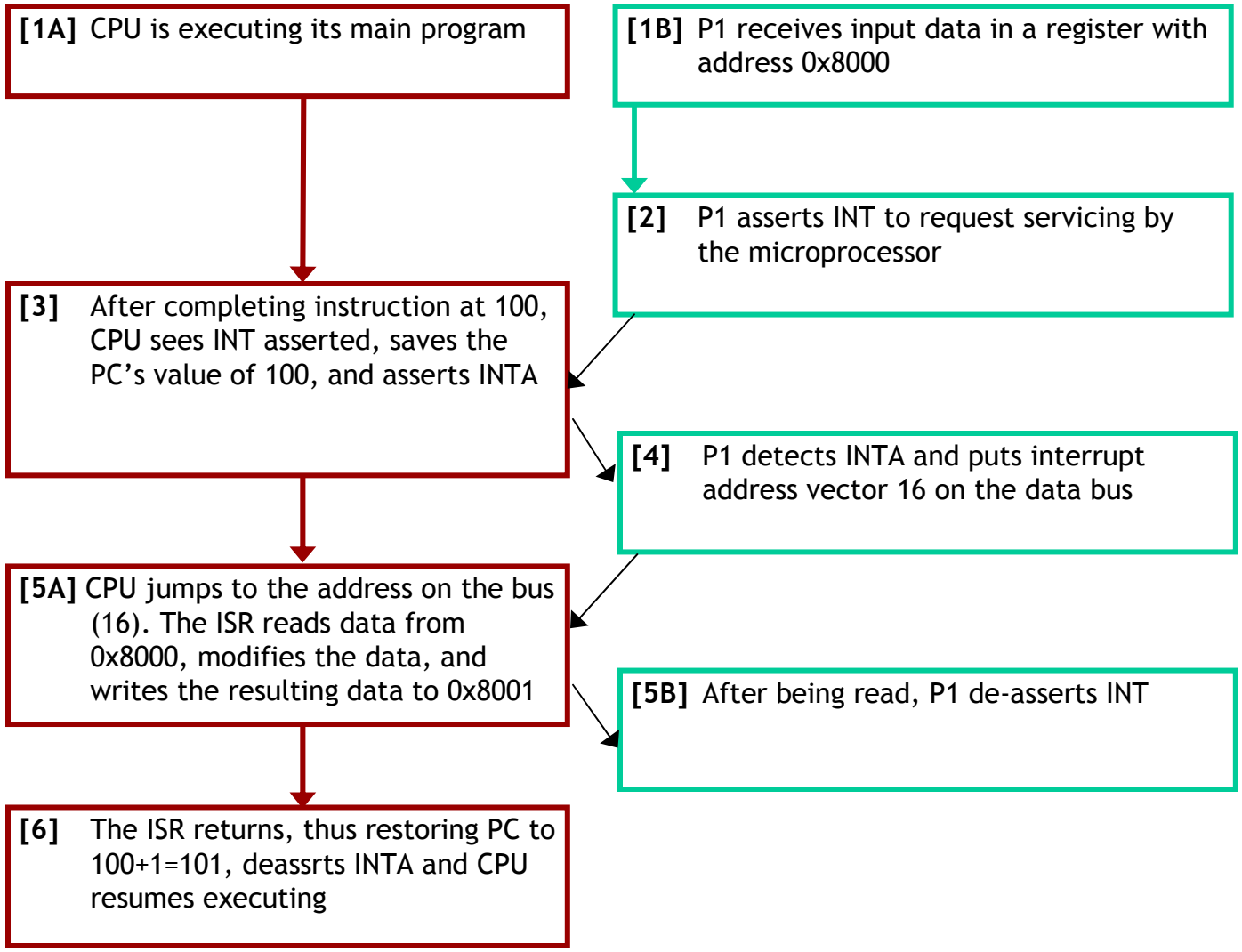
Fixed interrupt



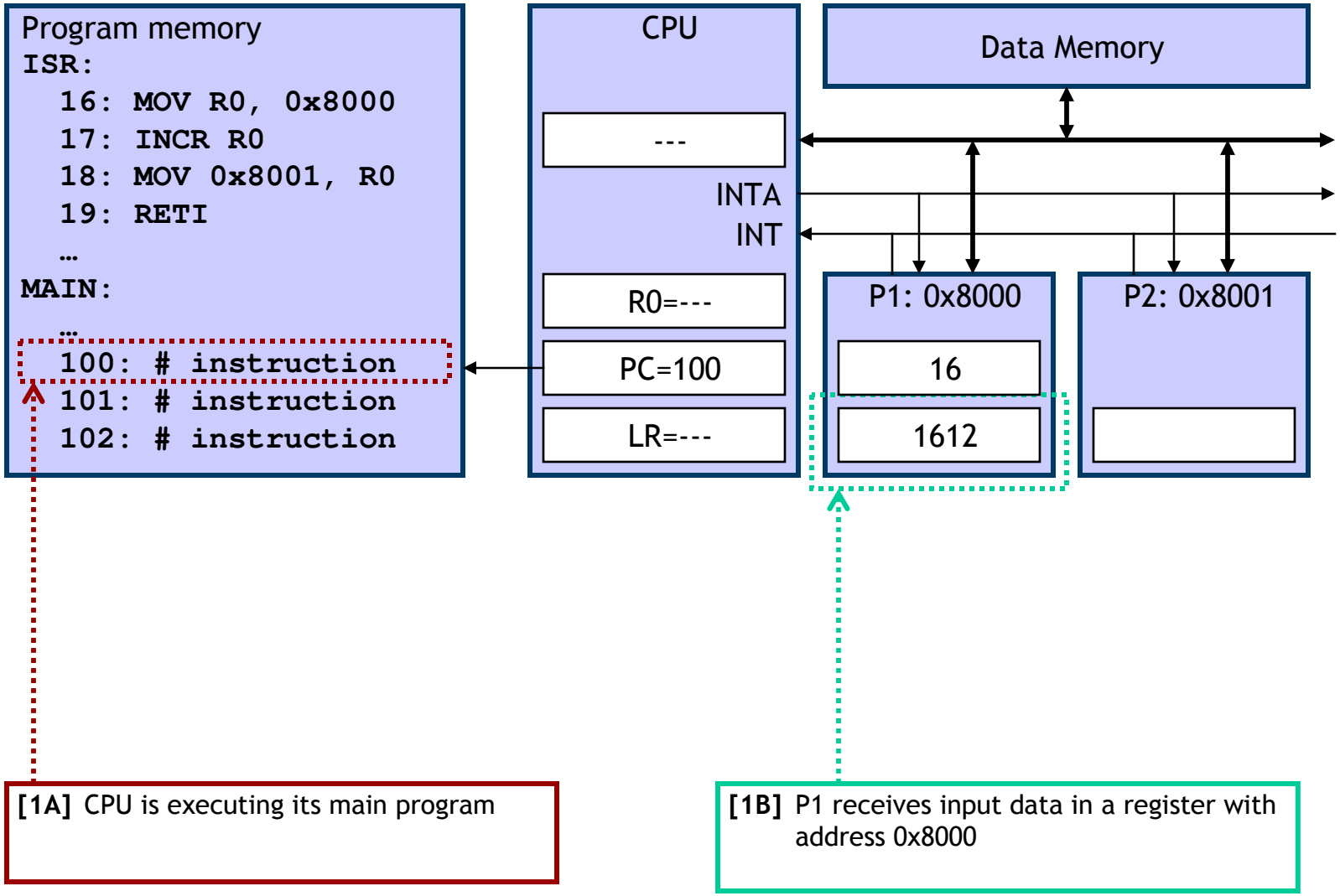
Fixed interrupt



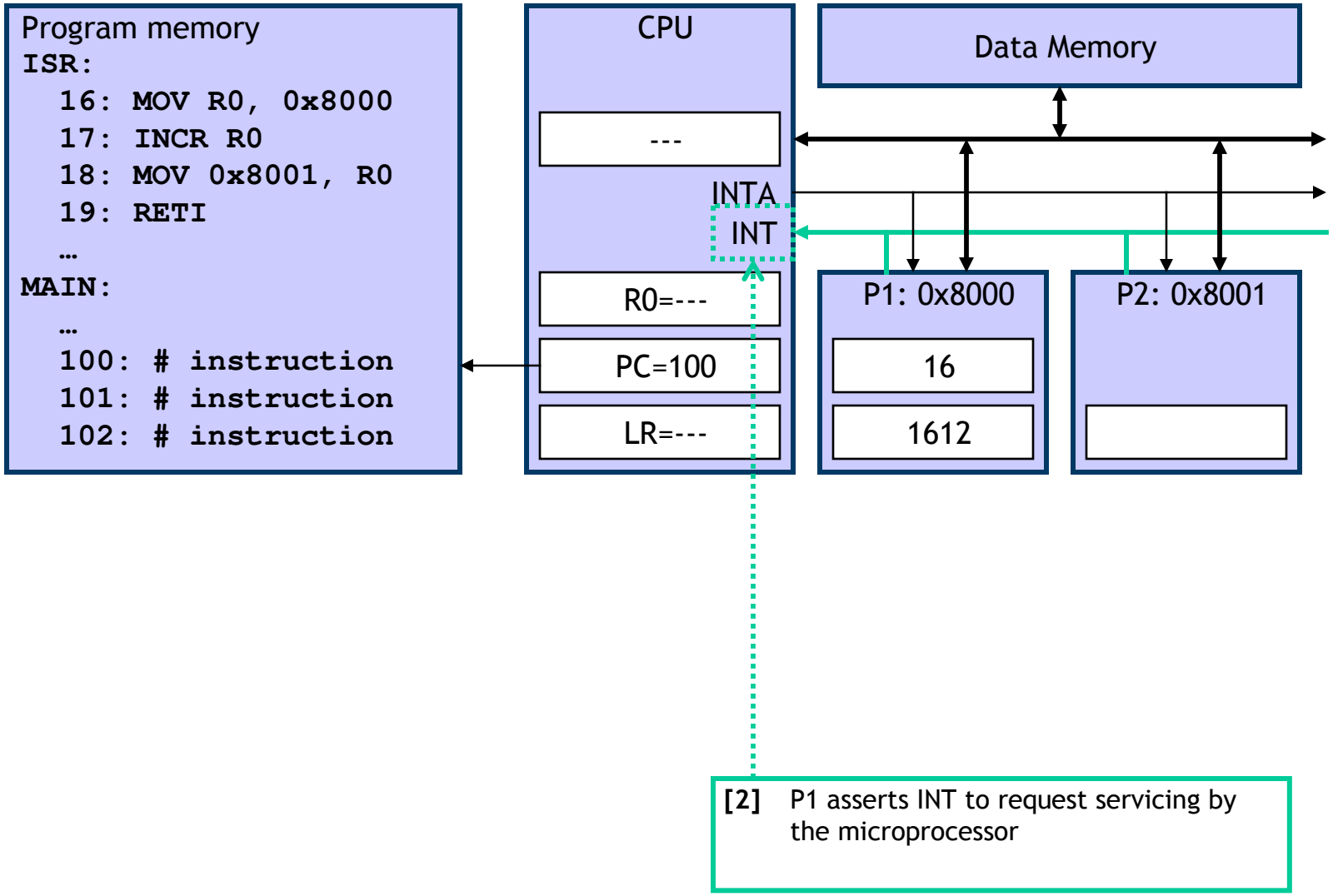
Vectored interrupt



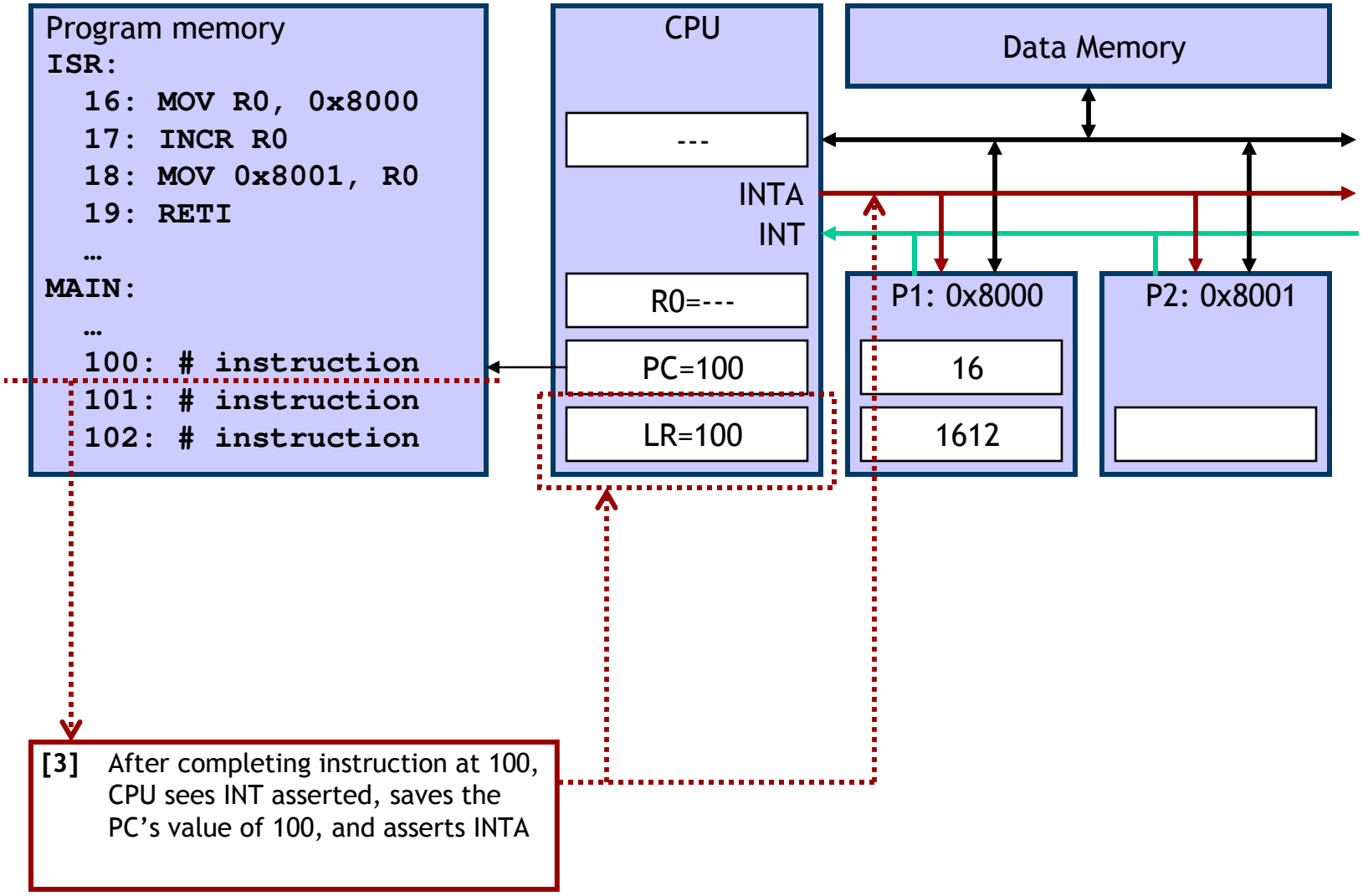
Vectored interrupt



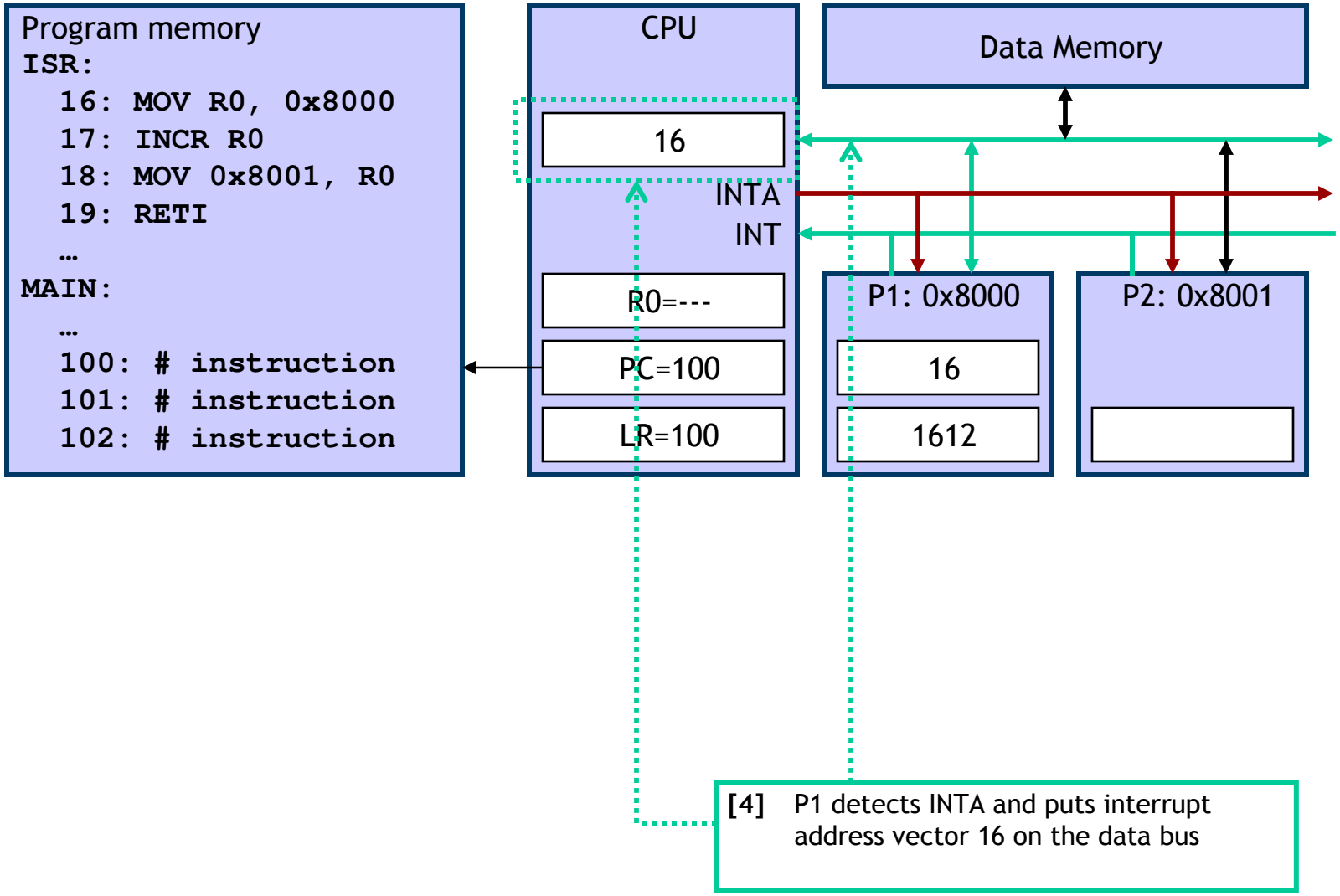
Vectored interrupt



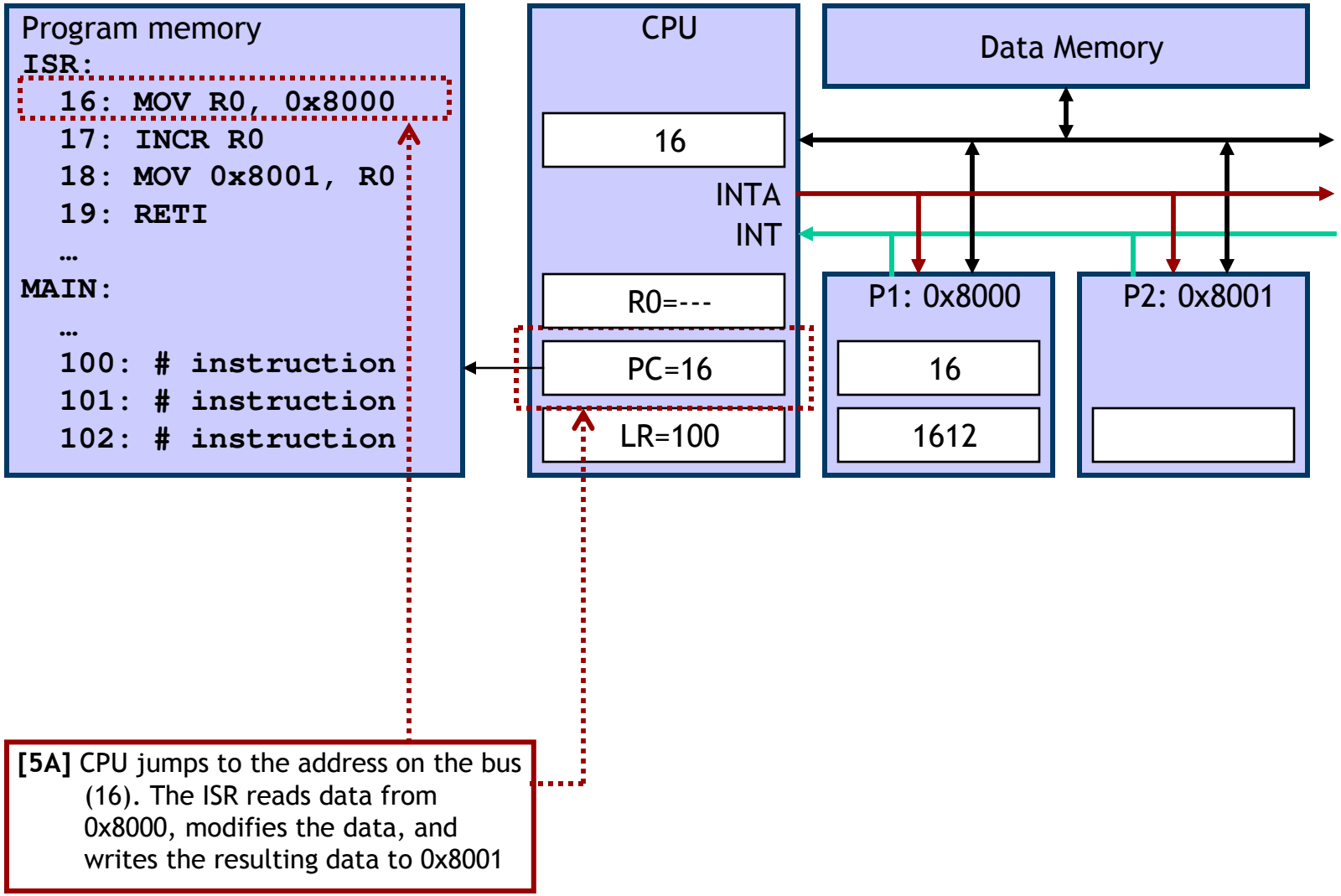
Vectored interrupt



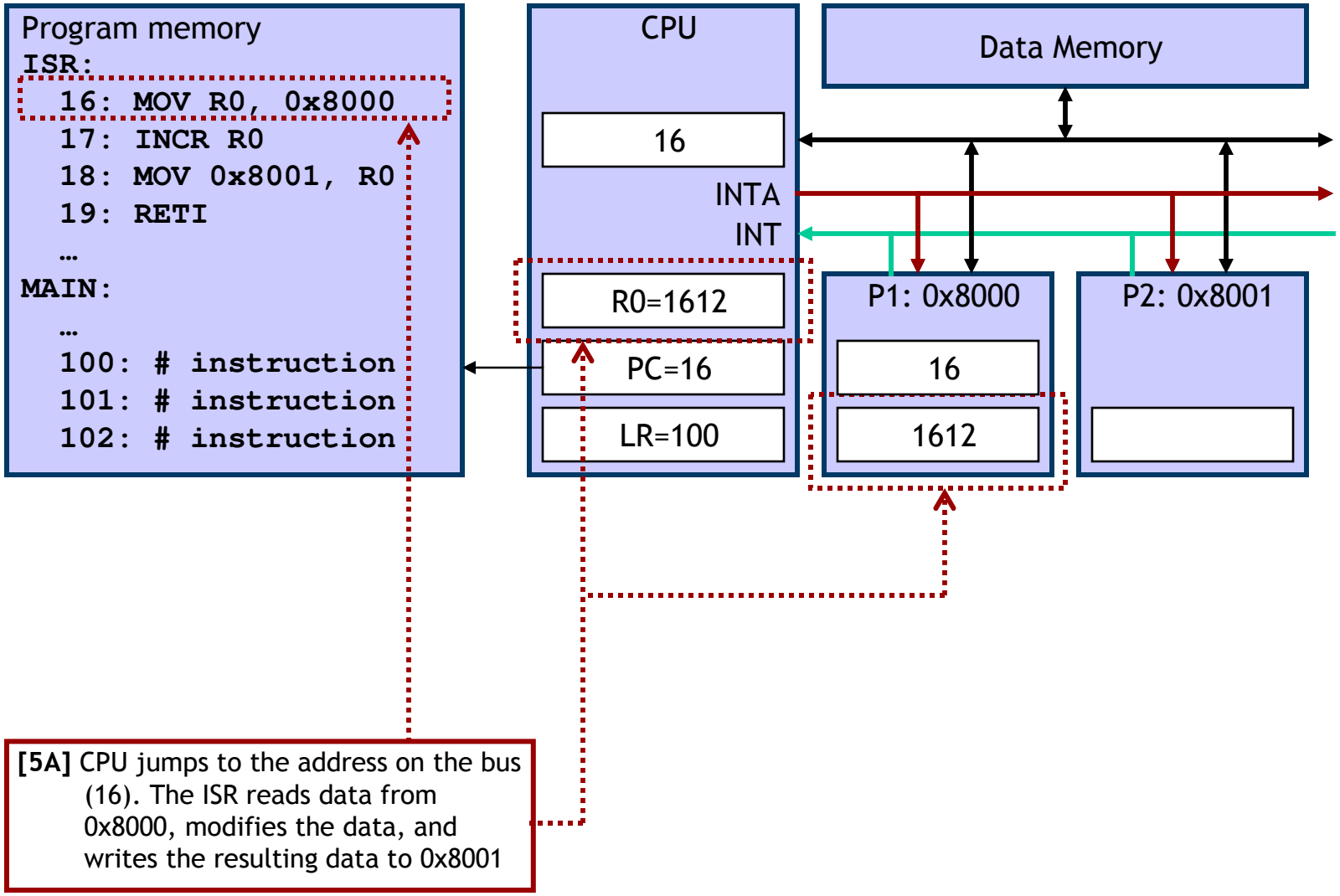
Vectored interrupt



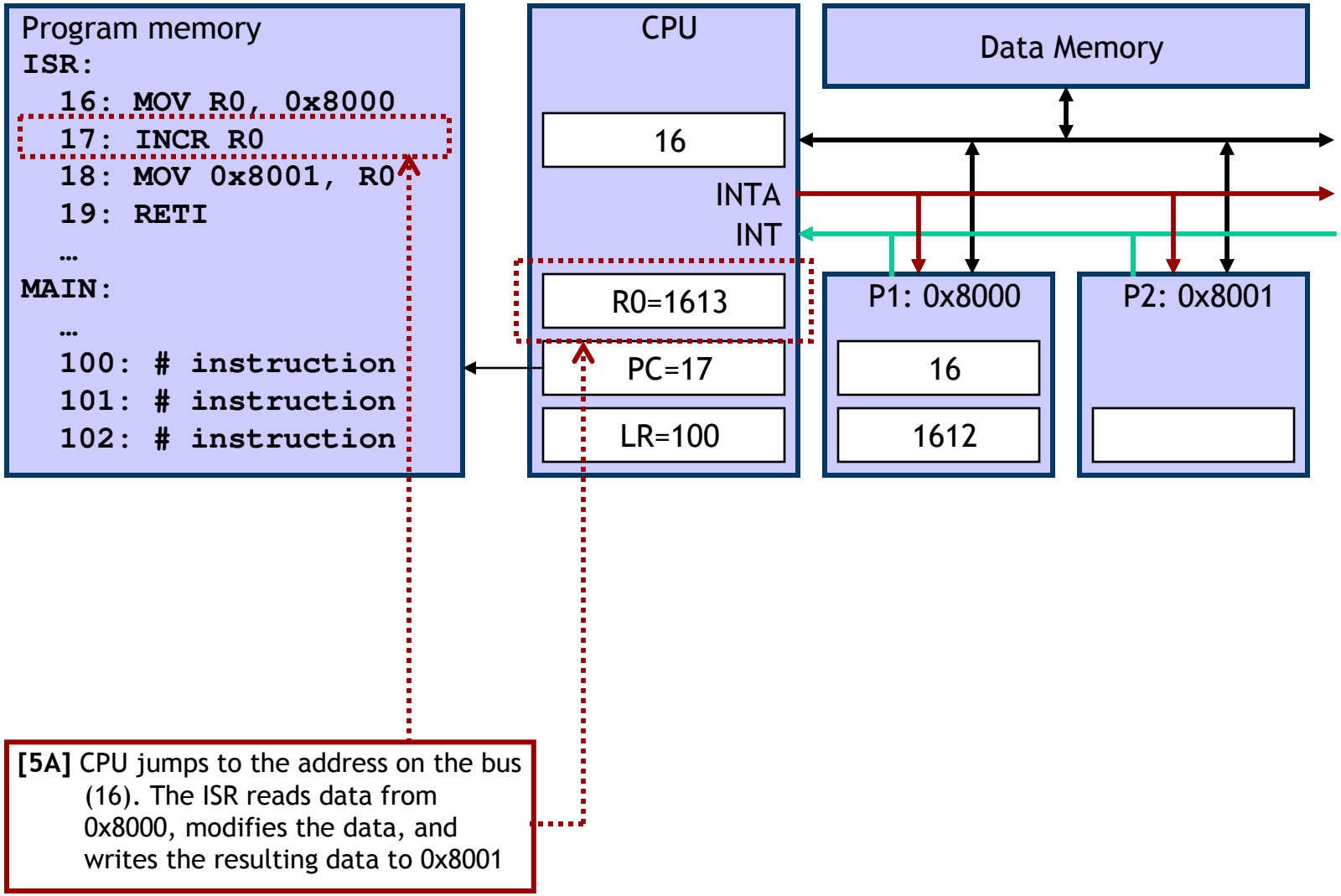
Vectored interrupt



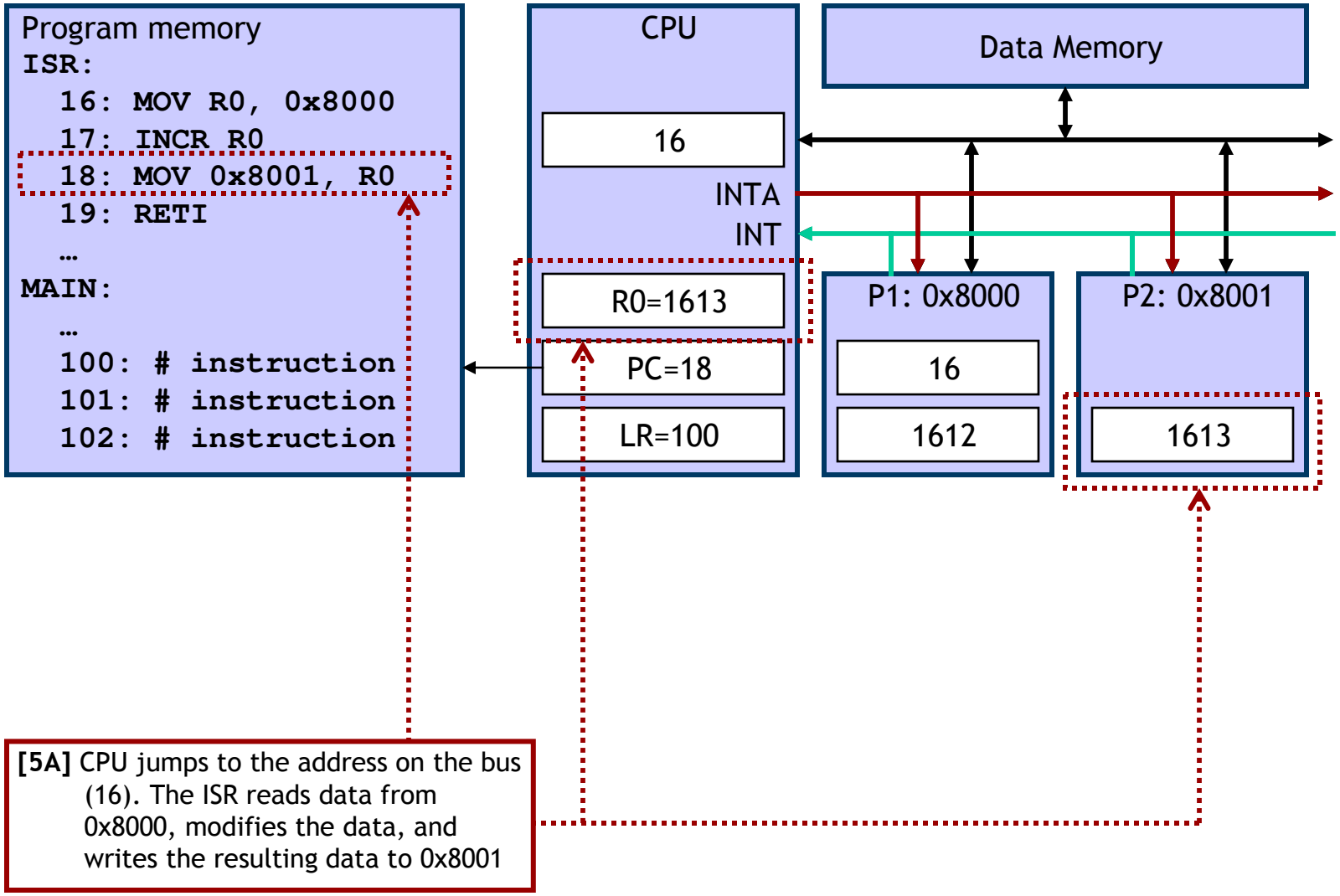
Vectored interrupt



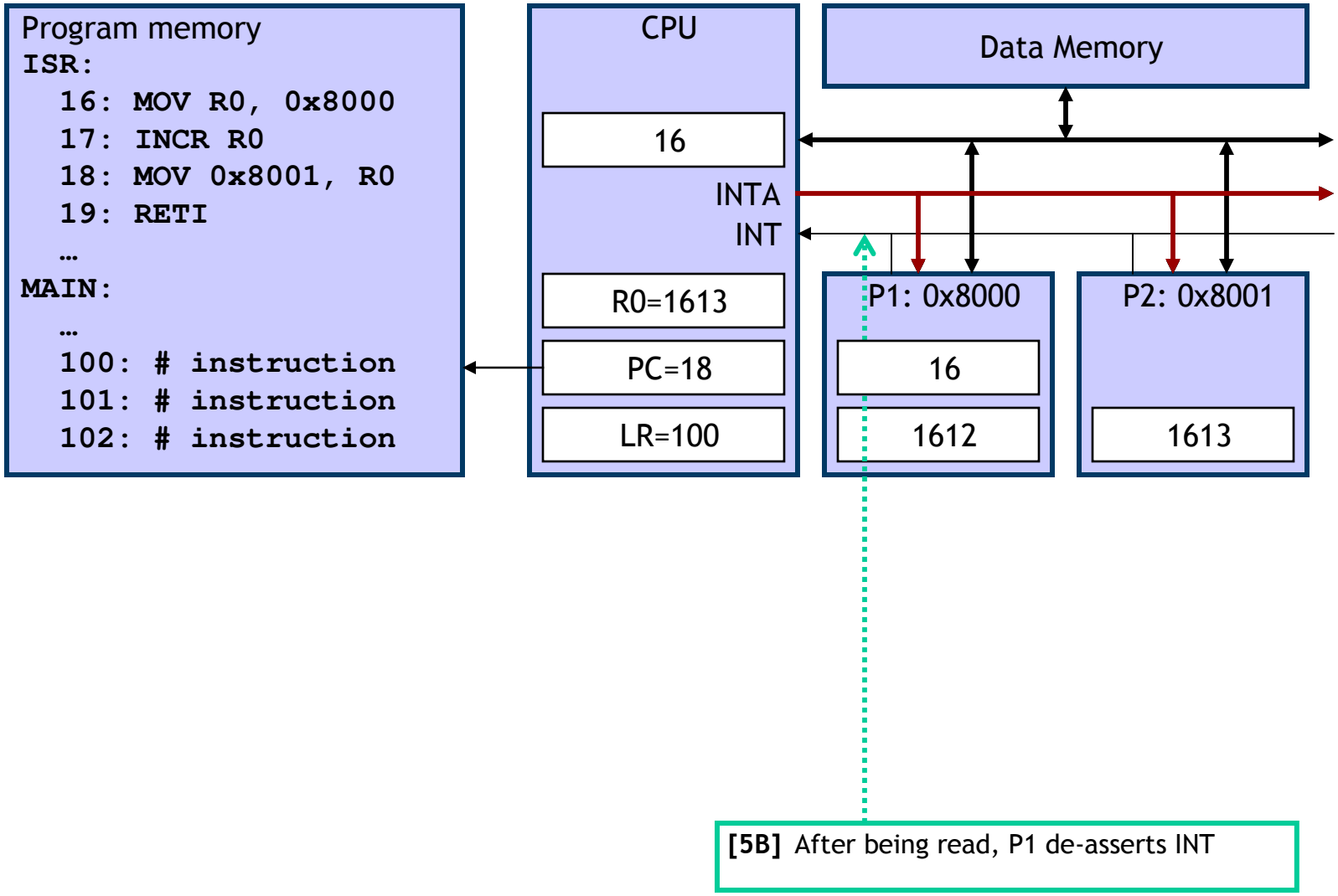
Vectored interrupt



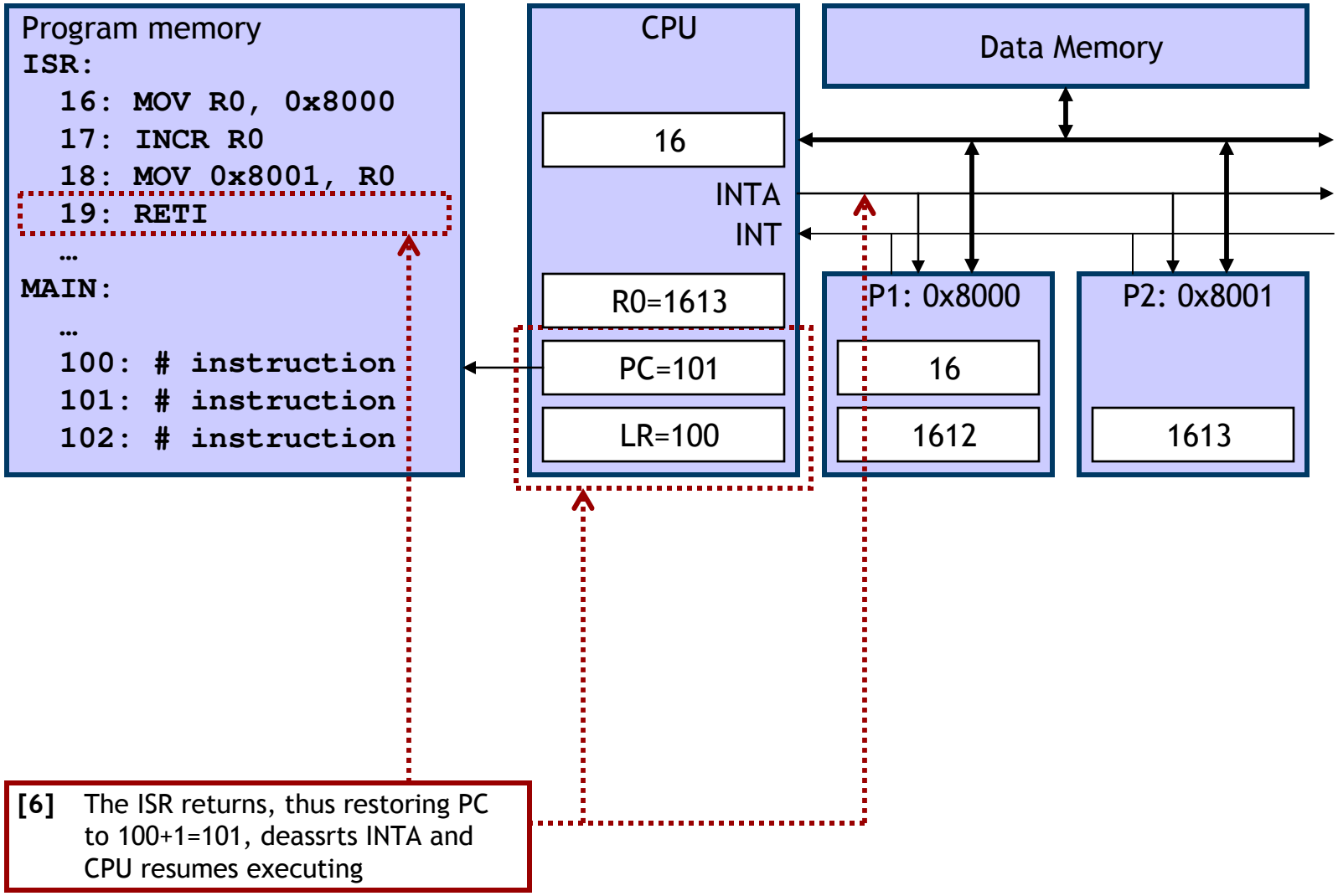
Vectored interrupt



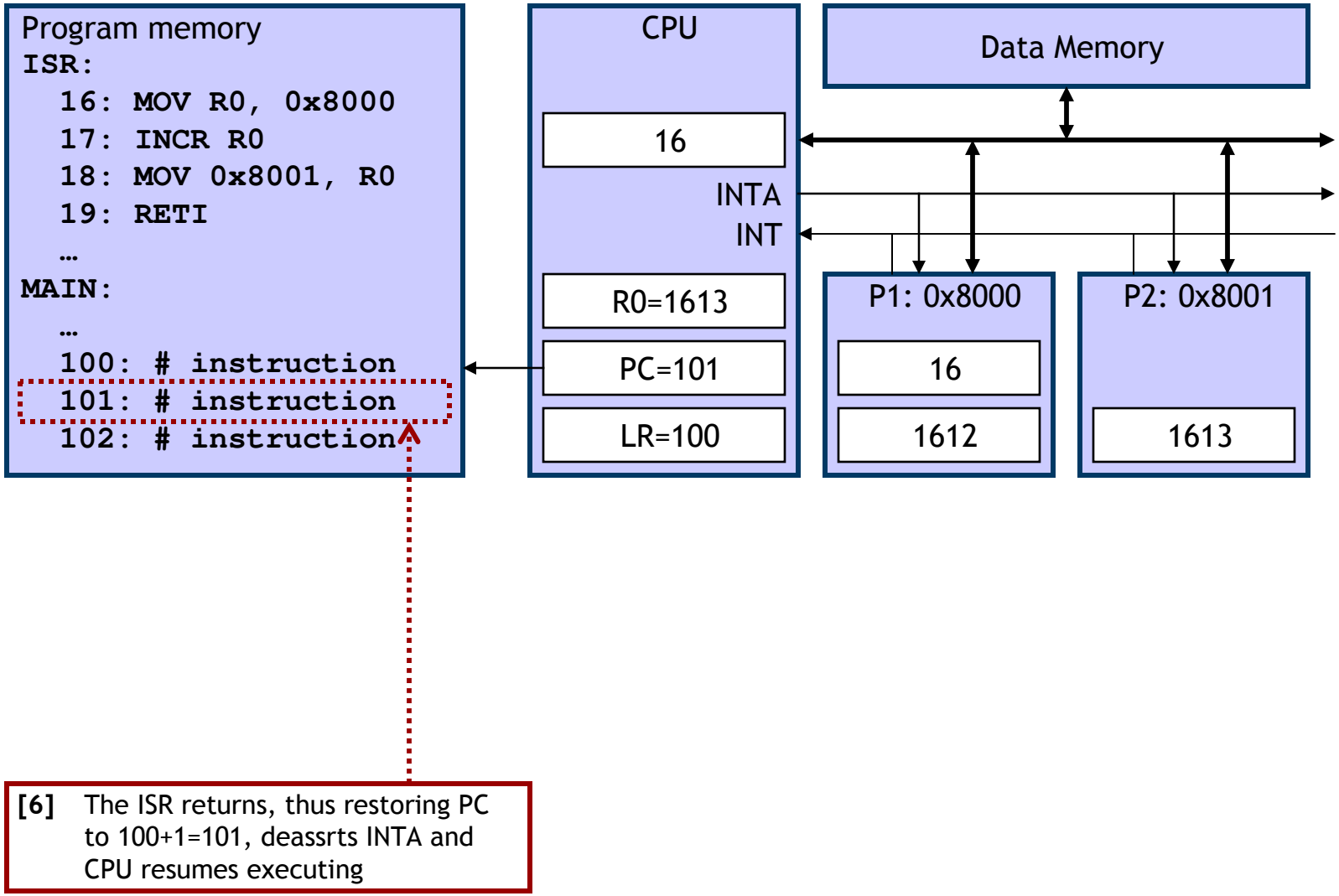
Vectored interrupt



Vectored interrupt



Vectored interrupt



DMA

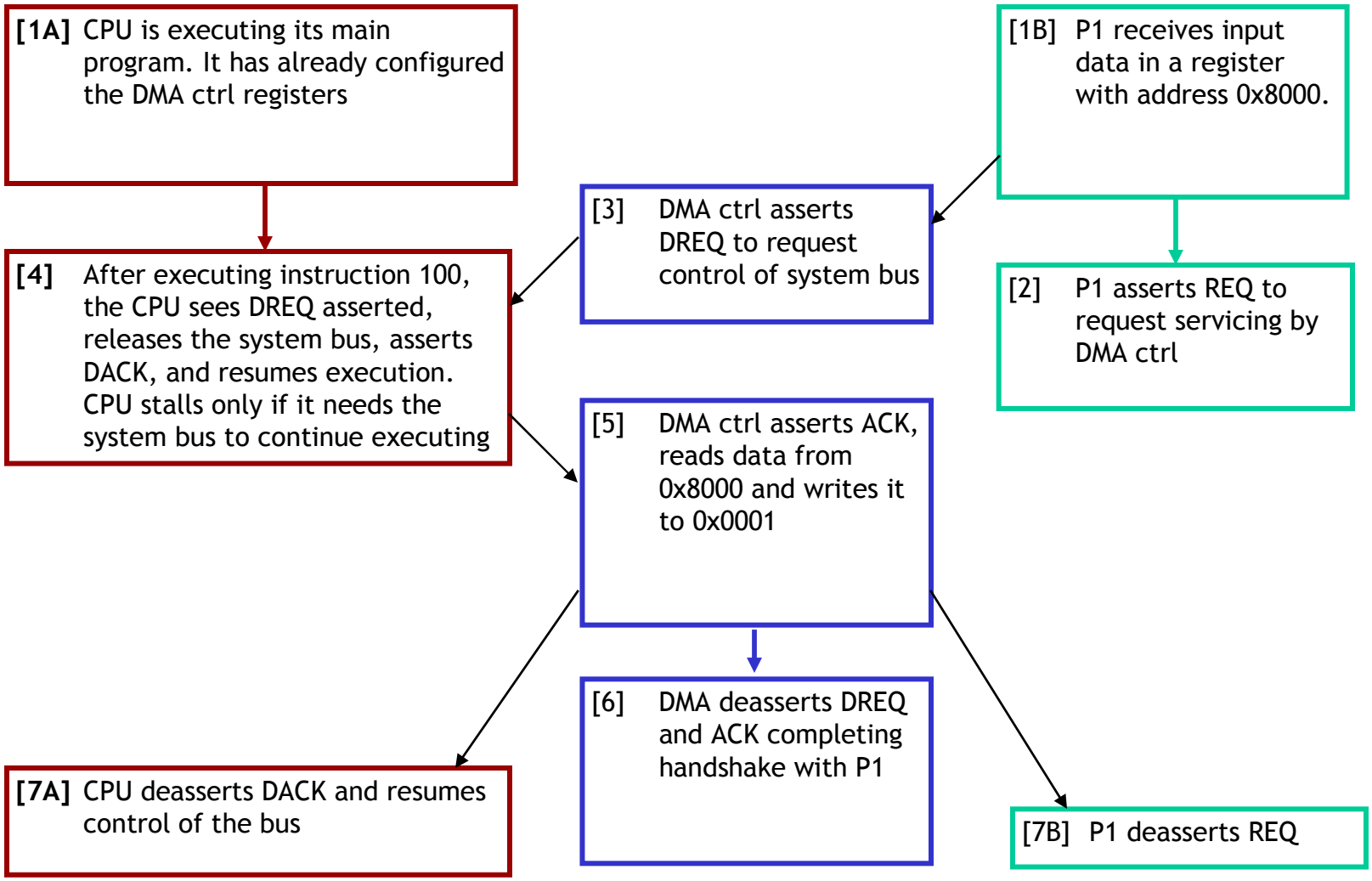
Il meccanismo degli interrupt

- Riduce i tempi di attesa dovuti ai cicli di polling
- Richiede comunque l'intervento del processore
  - Poco efficiente per trasferimenti di grandi quantità di dati tra periferiche e memoria centrali

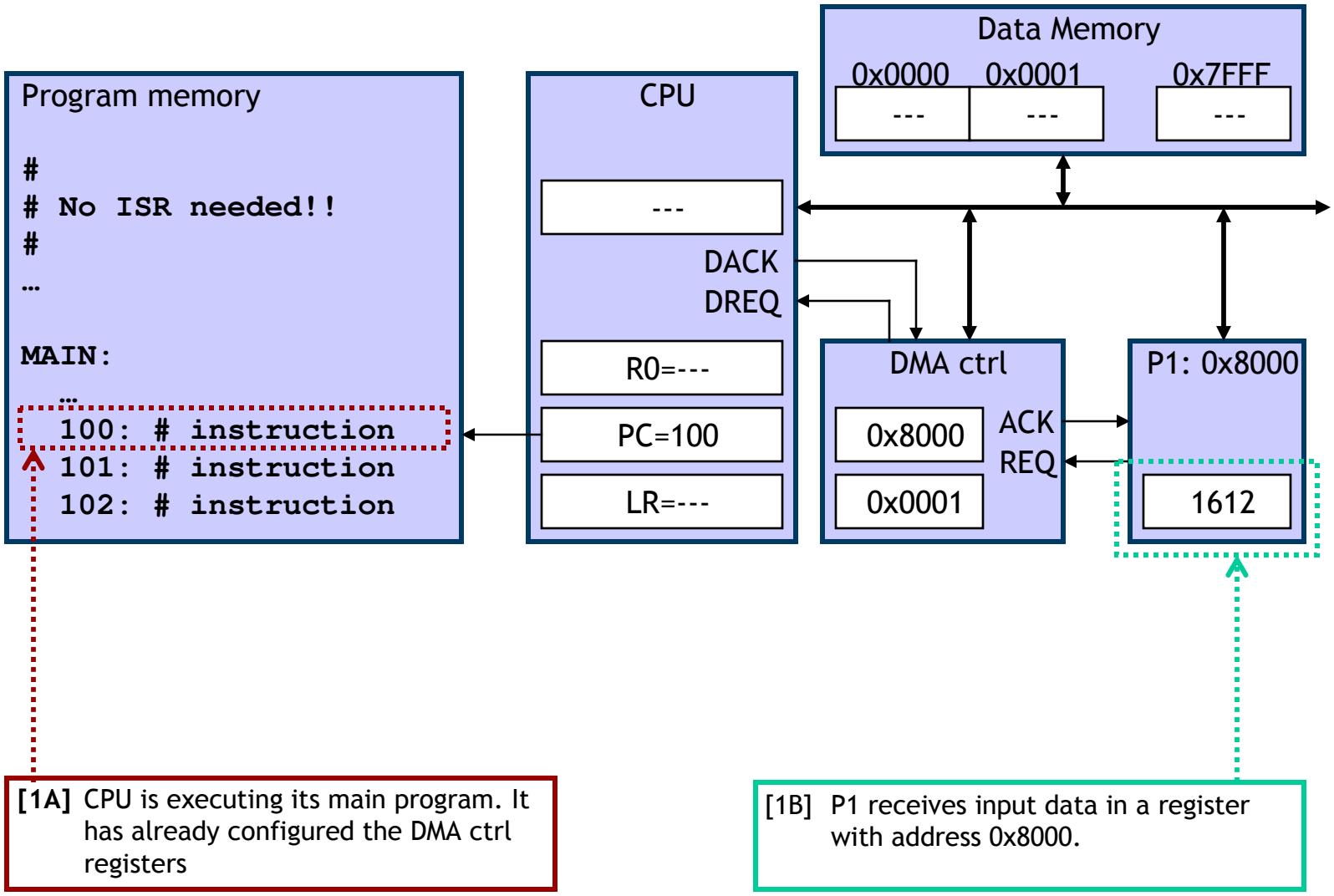
Direct memory access (DMA)

- Il trasferimento avviene direttamente tra dispositivi e memoria
- Il processore
  - Autorizzare la periferica ad accedere alla memoria
  - Programma uno speciale modulo (il DMA controller)
- In questo modo durante il trasferimento il processore può continuare l'esecuzione di un programma

DMA


[1A] CPU is executing its main program. It has already configured the DMA ctrl registers
[1B] P1 receives input data in a register with address 0x8000.
[2] P1 asserts REQ to request servicing by DMA ctrl
[3] DMA ctrl asserts DREQ to request control of system bus
[4] After executing instruction 100, the CPU sees DREQ asserted, releases the system bus, asserts DACK, and resumes execution. CPU stalls only if it needs the system bus to continue executing
[5] DMA ctrl asserts ACK, reads data from 0x8000 and writes it to 0x0001
[6] DMA deasserts DREQ and ACK completing handshake with P1
[7A] CPU deasserts DACK and resumes control of the bus
[7B] P1 deasserts REQ

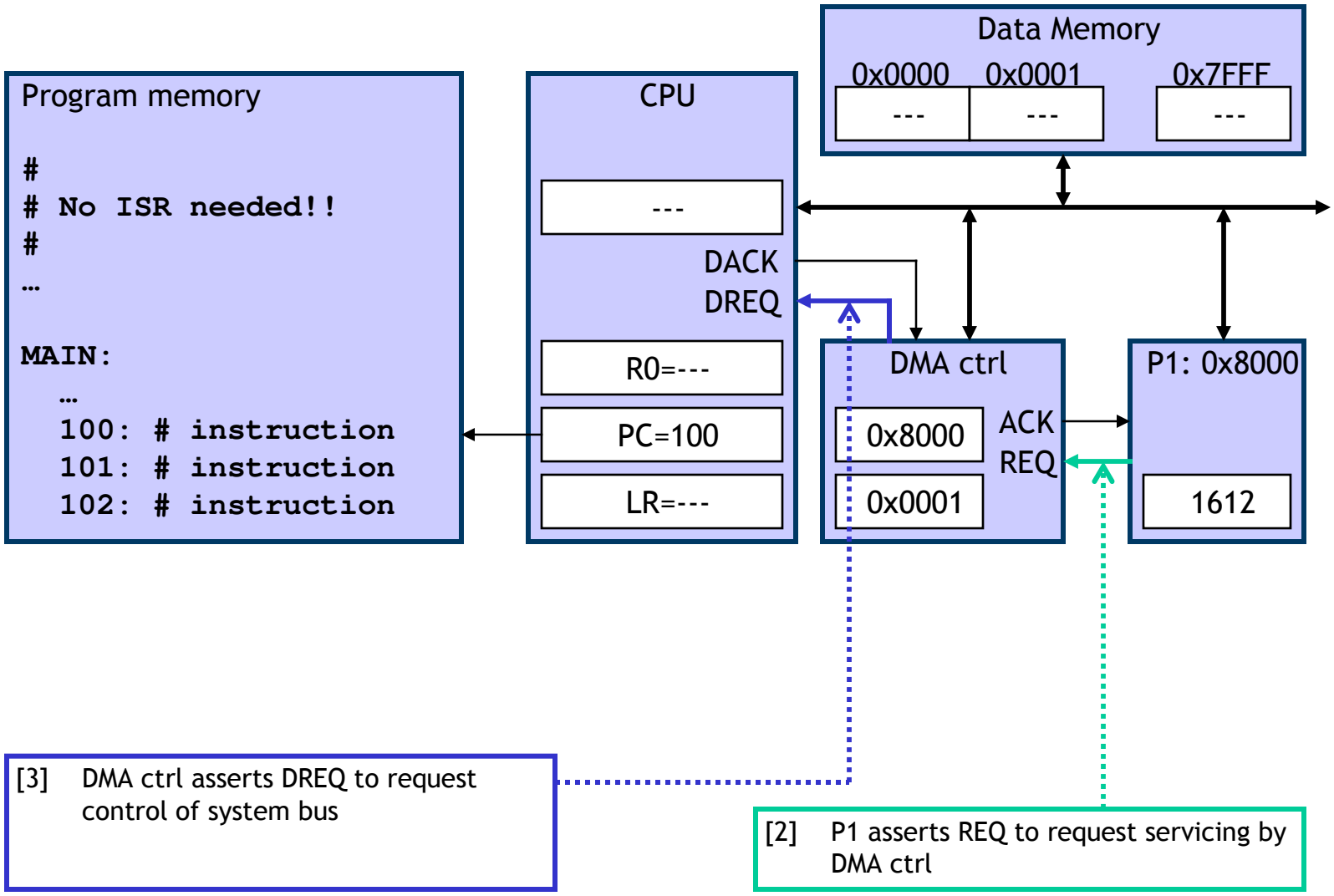
DMA


Data Memory
0x0000
0x0001
0x7FFF

Program memory
#
No ISR needed!!
#
…
MAIN:
…
100: # instruction
101: # instruction
102: # instruction
CPU

DACK
DREQ
R0=---
PC=100
LR=---
DMA ctrl
0x8000
0x0001
ACK
REQ
P1: 0x8000
1612
[1A] CPU is executing its main program. It has already configured the DMA ctrl registers
[1B] P1 receives input data in a register with address 0x8000.

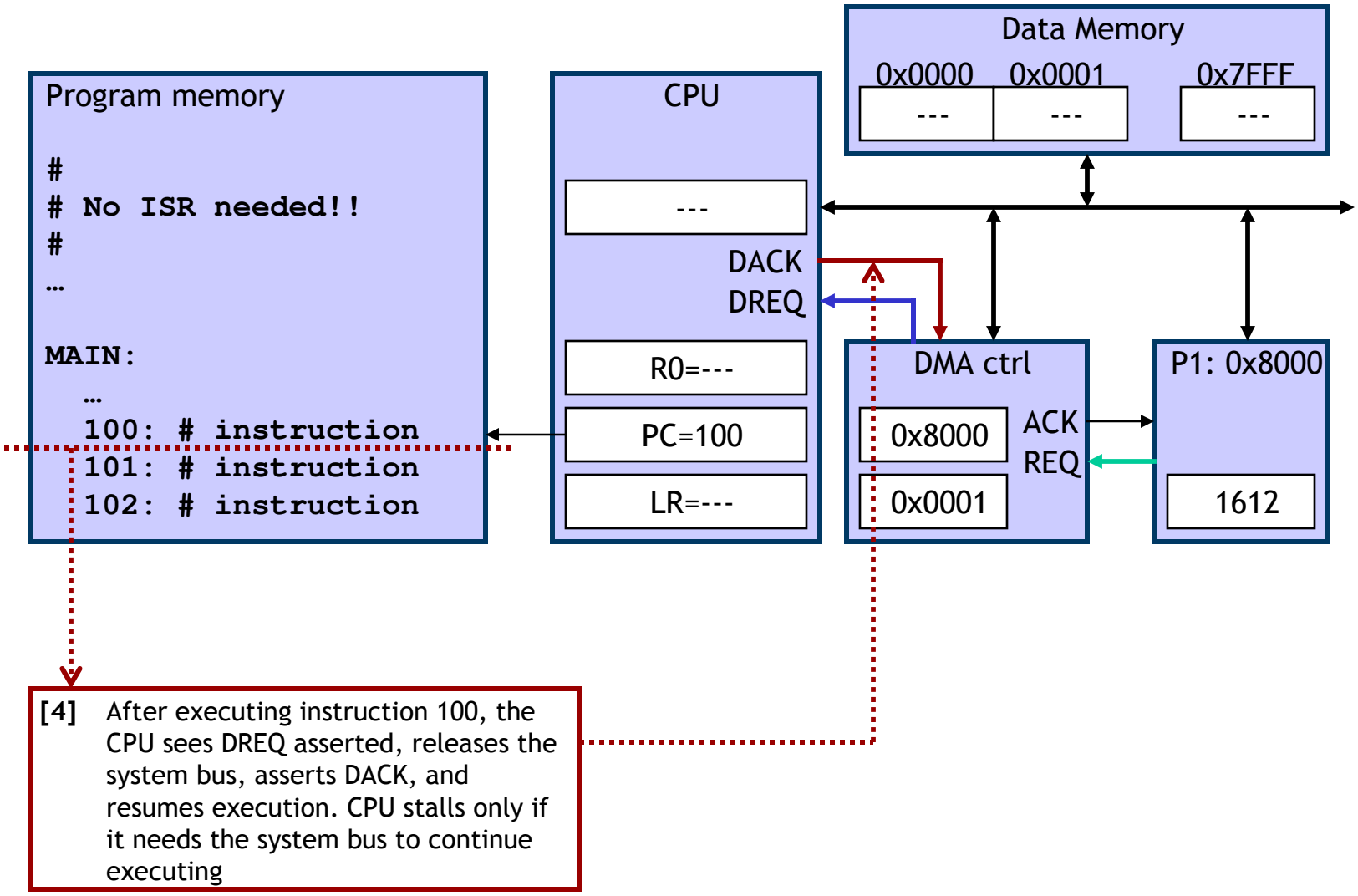
DMA


Program memory
#
No ISR needed!!
#
…
MAIN:
…
100: # instruction
101: # instruction
102: # instruction
CPU

DACK
DREQ
R0=---
PC=100
LR=---
Data Memory
0x0000
0x0001
0x7FFF

DMA ctrl
0x8000
0x0001
ACK
REQ
P1: 0x8000
1612
[3] DMA ctrl asserts DREQ to request control of system bus
[2] P1 asserts REQ to request servicing by DMA ctrl

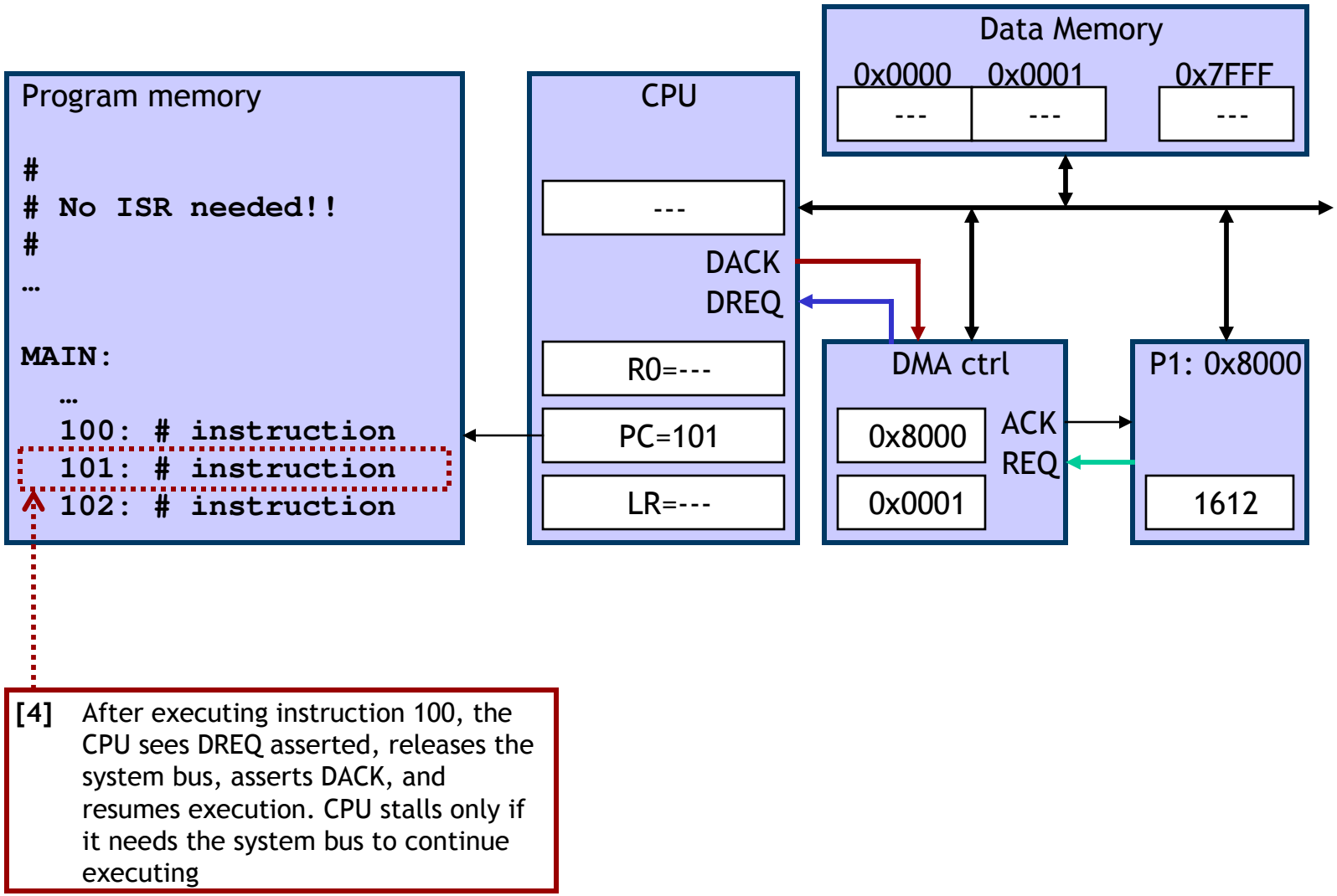
DMA


Program memory
#
No ISR needed!!
#
…
MAIN:
…
100: # instruction
101: # instruction
102: # instruction
CPU

DACK
DREQ
R0=---
PC=100
LR=---
Data Memory
0x0000
0x0001
0x7FFF

DMA ctrl
0x8000
0x0001
ACK
REQ
P1: 0x8000
1612
[4] After executing instruction 100, the CPU sees DREQ asserted, releases the system bus, asserts DACK, and resumes execution. CPU stalls only if it needs the system bus to continue executing

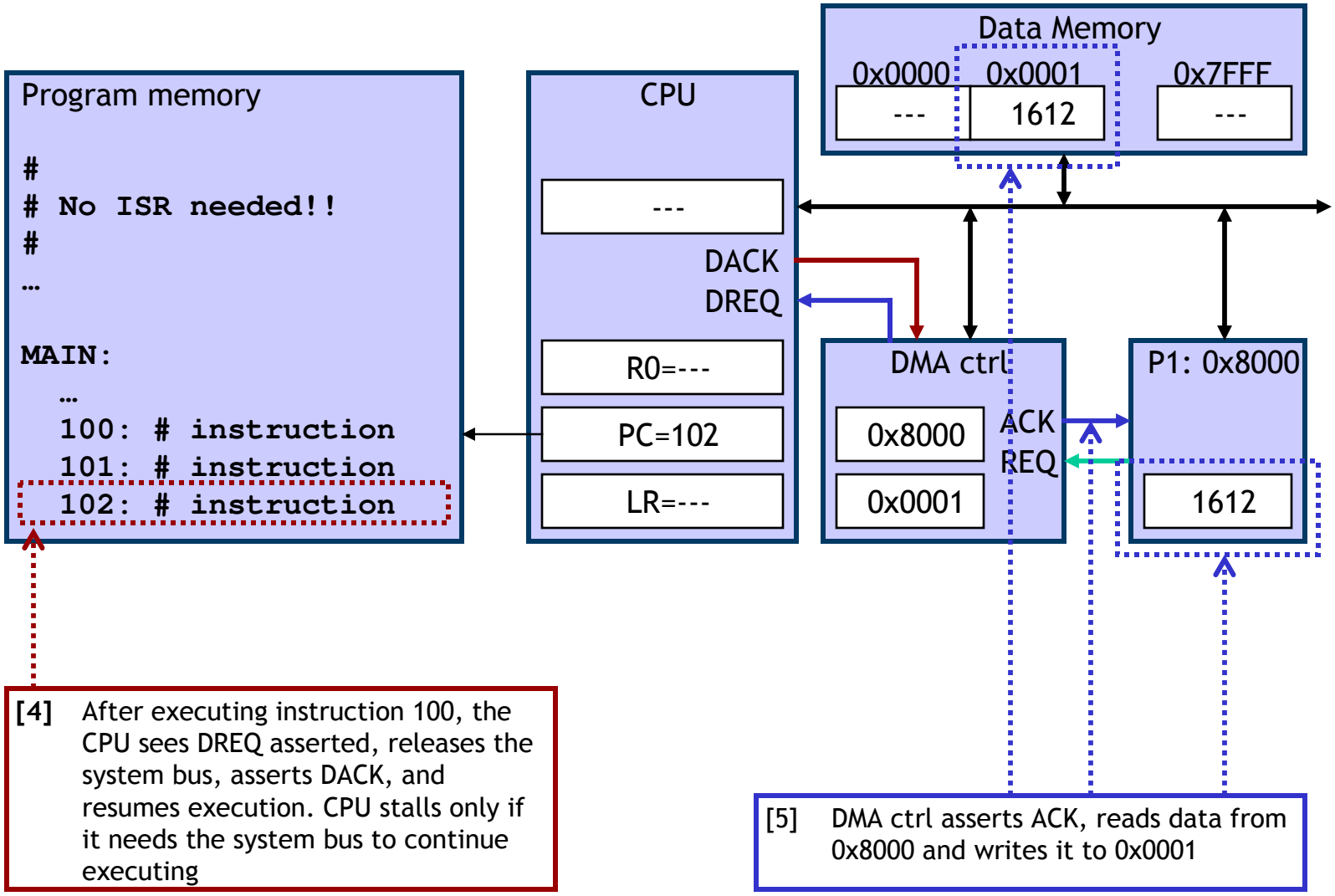
DMA


Program memory
#
No ISR needed!!
#
…
MAIN:
…
100: # instruction
101: # instruction
102: # instruction
CPU

DACK
DREQ
R0=---
PC=101
LR=---
Data Memory
0x0000
0x0001
0x7FFF

DMA ctrl
0x8000
0x0001
ACK
REQ
P1: 0x8000
1612
[4] After executing instruction 100, the CPU sees DREQ asserted, releases the system bus, asserts DACK, and resumes execution. CPU stalls only if it needs the system bus to continue executing

DMA

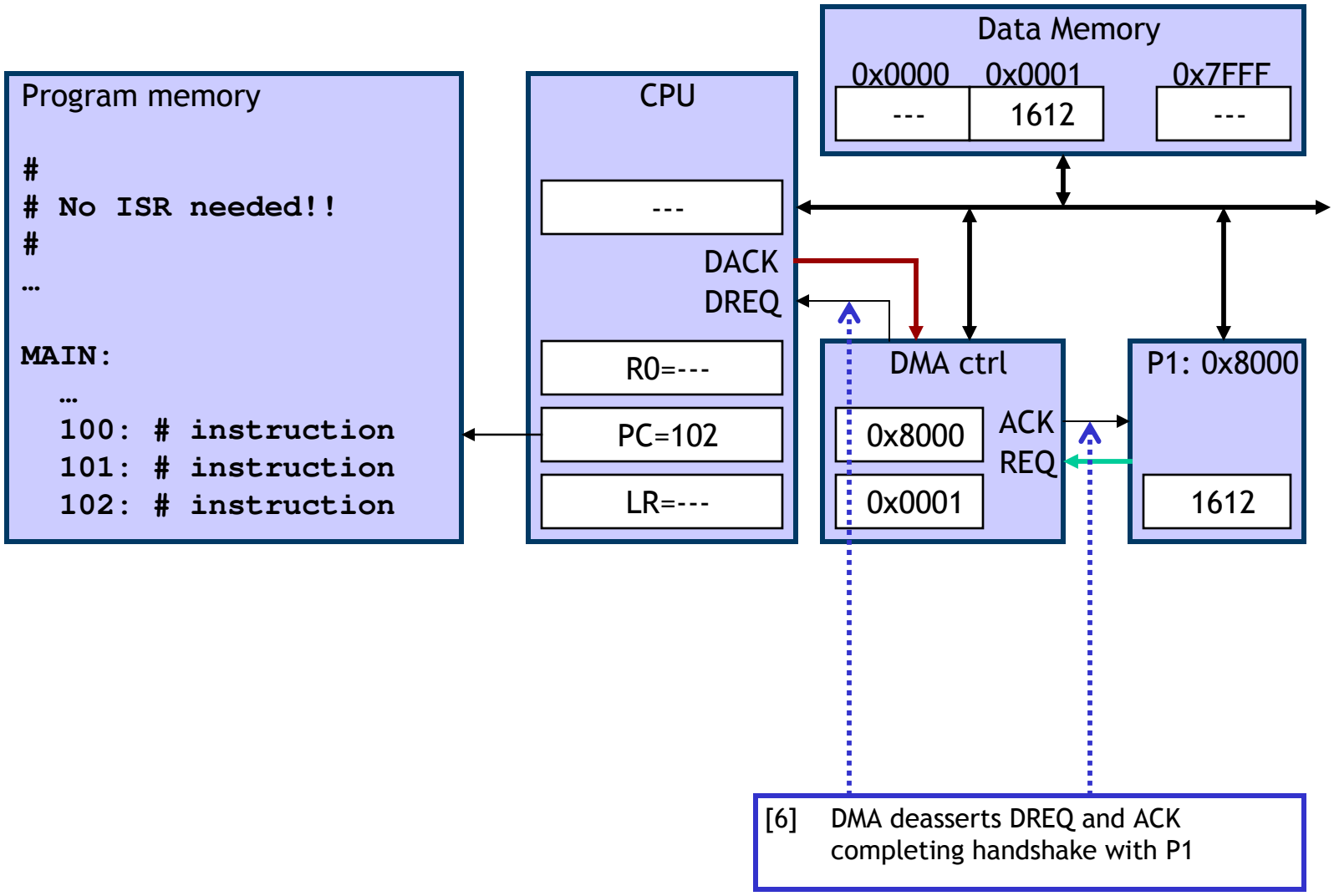

Program memory
#
No ISR needed!!
#
…
MAIN:
…
100: # instruction
101: # instruction
102: # instruction
CPU

DACK
DREQ
R0=---
PC=102
LR=---
Data Memory
0x0000
0x0001
0x7FFF

1612

DMA ctrl
0x8000
0x0001
ACK
REQ
P1: 0x8000
1612
[4] After executing instruction 100, the CPU sees DREQ asserted, releases the system bus, asserts DACK, and resumes execution. CPU stalls only if it needs the system bus to continue executing
[5] DMA ctrl asserts ACK, reads data from 0x8000 and writes it to 0x0001

DMA

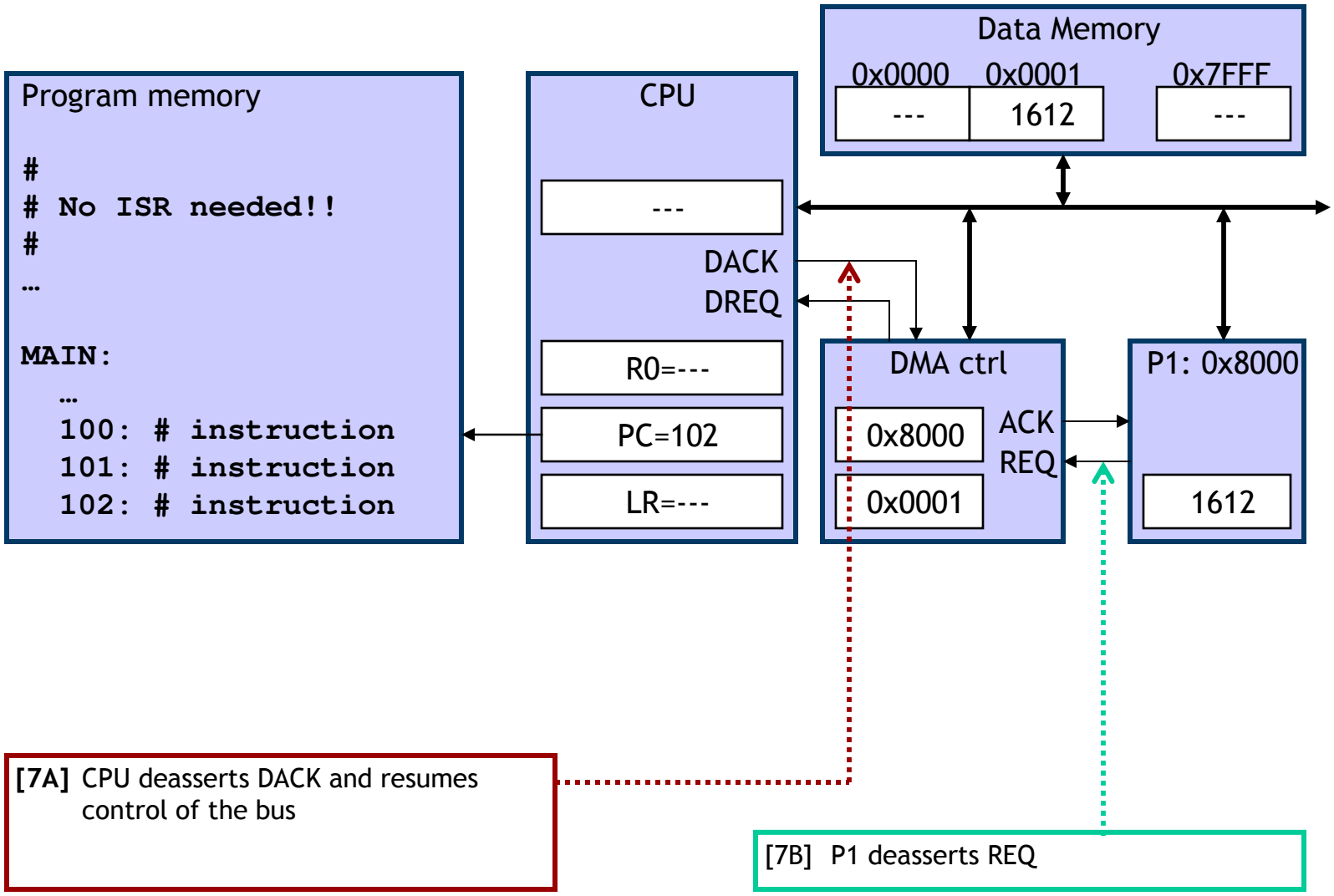

Program memory
#
No ISR needed!!
#
…
MAIN:
…
100: # instruction
101: # instruction
102: # instruction
CPU

DACK
DREQ
R0=---
PC=102
LR=---
Data Memory
0x0000
0x0001
0x7FFF

1612

DMA ctrl
0x8000
0x0001
ACK
REQ
P1: 0x8000
1612
[6] DMA deasserts DREQ and ACK completing handshake with P1

DMA


Program memory
#
No ISR needed!!
#
…
MAIN:
…
100: # instruction
101: # instruction
102: # instruction
CPU

DACK
DREQ
R0=---
PC=102
LR=---
Data Memory
0x0000
0x0001
0x7FFF

1612

DMA ctrl
0x8000
0x0001
ACK
REQ
P1: 0x8000
1612
[7A] CPU deasserts DACK and resumes control of the bus
[7B] P1 deasserts REQ

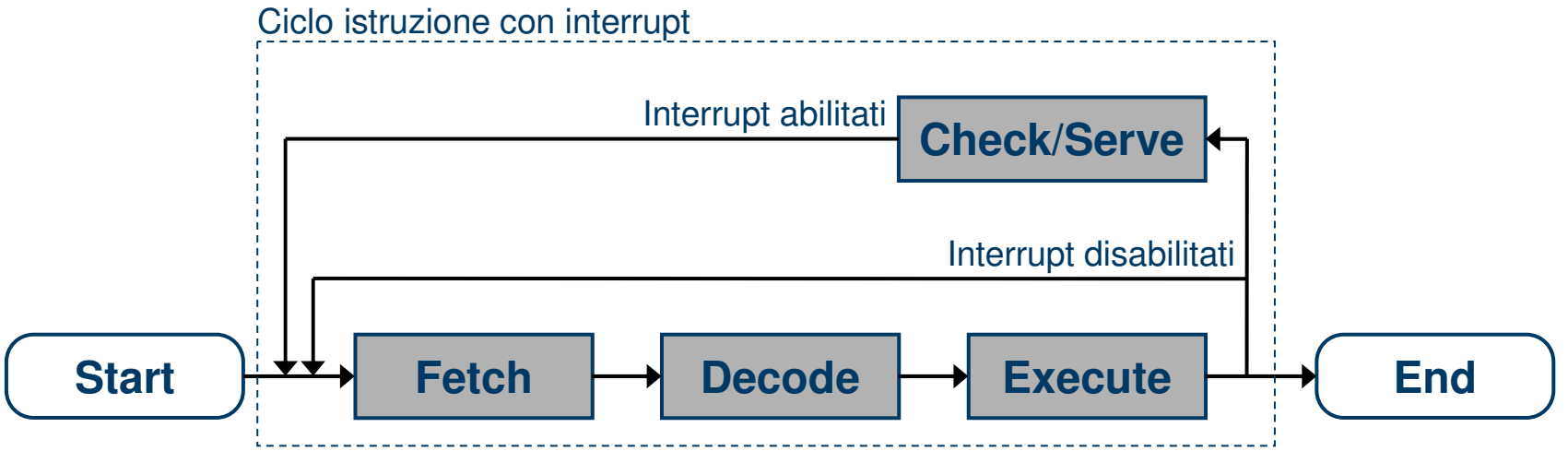
Ciclo istruzione modificato

Per gestire interrupt e/o DMA

- È necessario modificare il ciclo istruzione visto in precedenza

Al termine dell'esecuzione di una istruzione

- Si verifica se gli interrupt sono abilitati
- Se sono abilitati, se ne verifica la presenza



Flusso di esecuzione

Interrupt singolo

- Il processore serve l'interrupt appena lo riceve

Interrupt multipli

- Il processore riceve un interrupt mentre sta eseguendo la ISR di un interrupt precedente

Due soluzioni

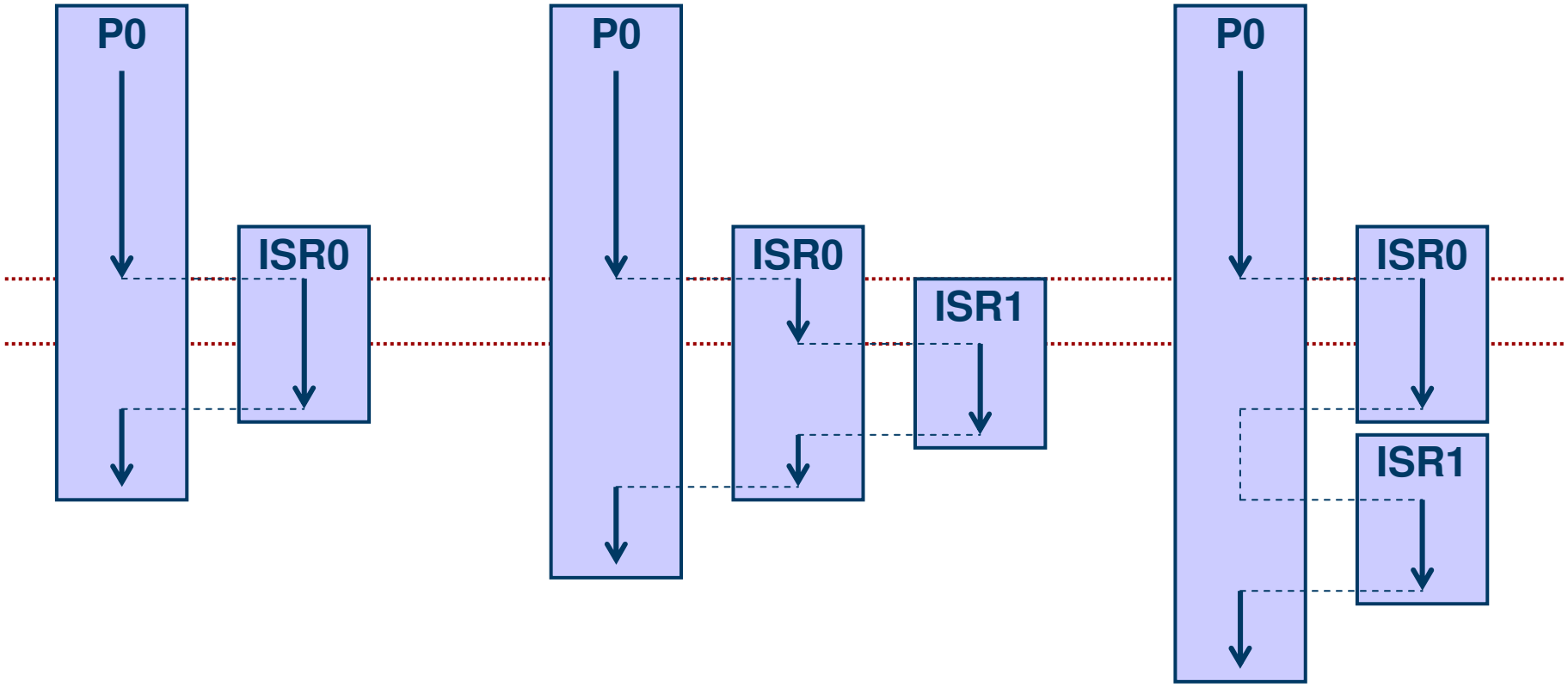
- Interrupt annidato
  - Gli interrupt rimangono abilitati
  - Il processore interrompe la ISR e serve il secondo interrupt
- Interrupt differito
  - Gli interrupt vengono disabilitati durante una ISR
  - Le richieste vengono accodate
  - Il processore serve il secondo interrupt dopo aver concluso la ISR del primo

Flusso di esecuzione

Interrupt semplice

Interrupt annidato

Interrupt differito



Flusso di esecuzione

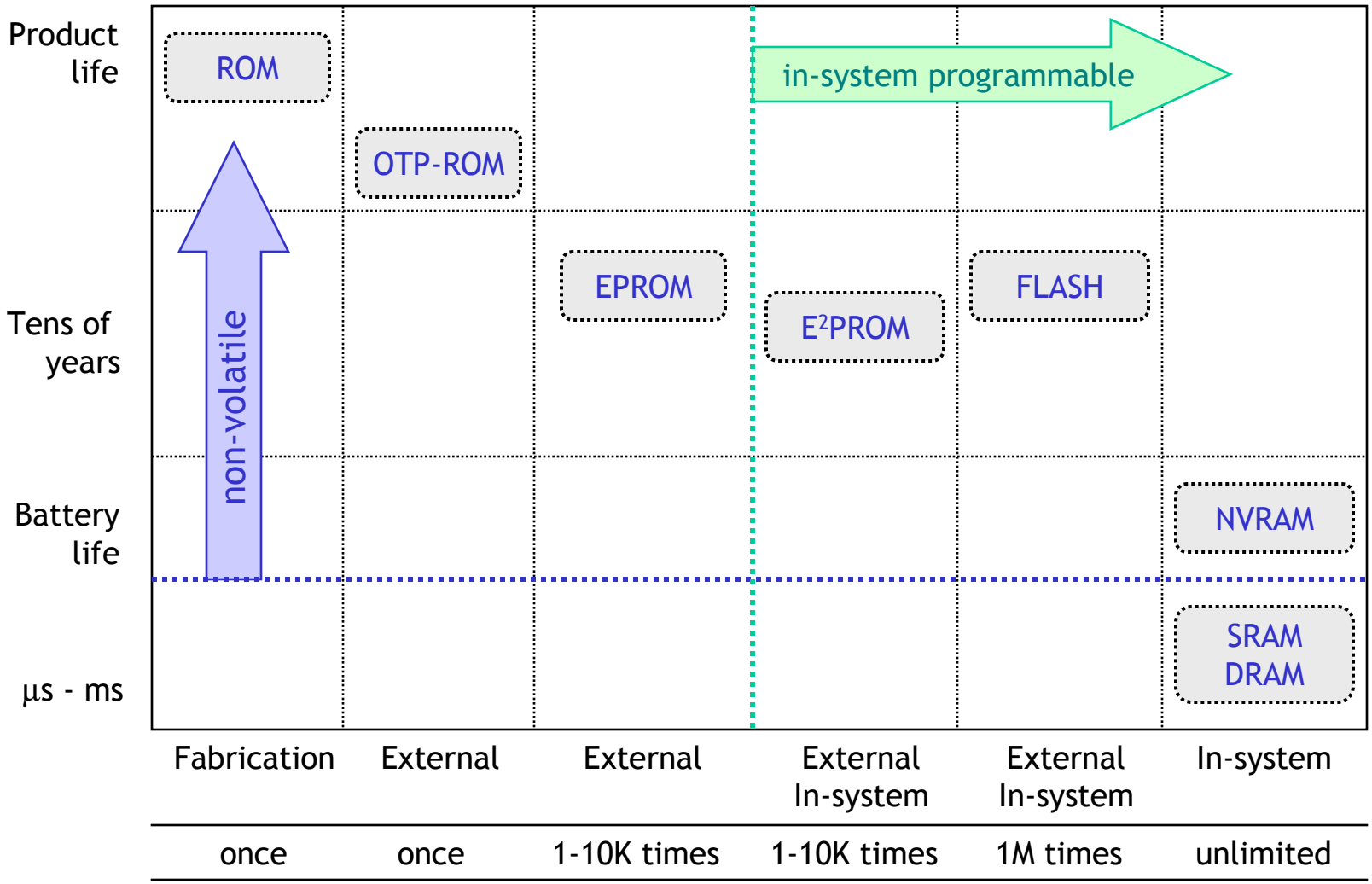
È possibile e utile adottare una trategia mista con

- Interrupt annidati
- Interrupt differiti

Si definiscono livelli di priorità

- Un interrupt di priorità più alta
  - Può interrompere la ISR di un interrupt di priorità minore
- Un interrupt di bassa priorità
  - Viene differito fino alla conclusione della ISR in esecuzione

Memorie


Product life
Tens of years
Battery life
µs - ms
ROM
OTP-ROM
EPROM
E²PROM
FLASH
NVRAM
SRAM
DRAM
in-system programmable
non-volatile
Fabrication
External
External
External In-system
External In-system
In-system
once
once
1-10K times
1-10K times
1M times
unlimited

Gerarchia di memoria

Registri

- Molto piccoli, estremamente veloci

Cache L0

- Piccola, molto veloce
- Molto costosa

Cache L1

- Medie dimensioni, veloce

Memoria principale

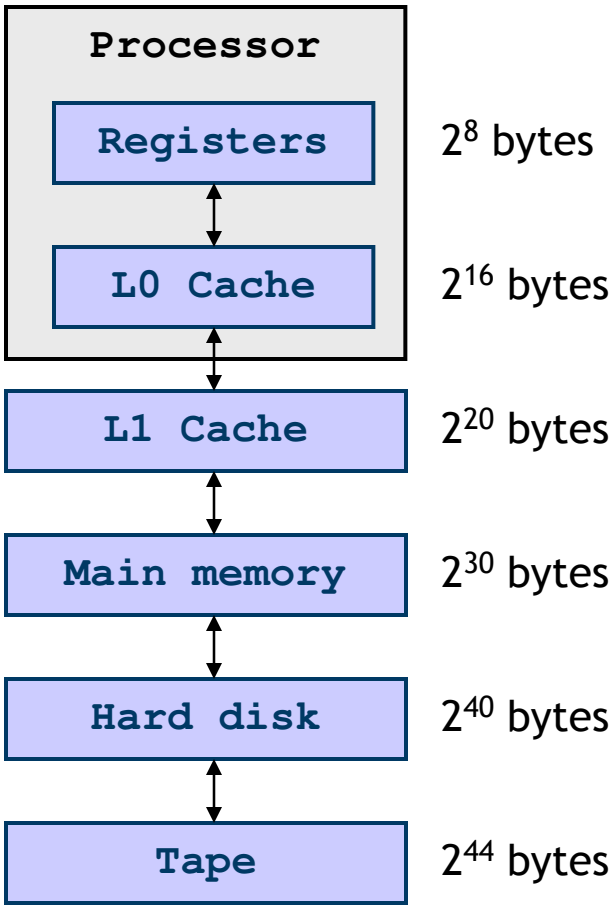
- Grande, non molto veloce
- Poco costosa

Hard disk

- Molto grande, lenta

Tape

- Enorme, lentissima



Memorie cache

Aumenta le prestazioni di un sistema di calcolo

- Il processore è più veloce della memoria

I dati letti/scritti dalla memoria godono di due proprietà

- Località spaziale
  - Due accessi successivi hanno una elevata probabilità di avvenire ad indirizzi vicini nello spazio
- Località temporale
  - Lo stesso indirizzo ha una elevata probailità di essere acceduto due o più volte in un breve intervallo di tempo

Ne consegue che

- L'area di memoria acceduta da un programma varia molto lentamente nel tempo
- Una memoria piccola può contenere una porzione della memoria centrale che sarà usata con alta frequenza

Cache mapping

La cache è molto più piccola della memoria centrale

- Associare gli indirizzi in cache a quelli in memoria centrale
- Determinare se un dato indirizzo è disponibile in cache o no
  - Se è disponibile si ha un cache hit
  - Se non è dispoibile si ha un cache miss

Tre tecniche di base

- Mapping diretto
- Mapping completamente associativo
- Mapping set-associativo

Tali tecniche presentano

- Prestazioni differenti
- Costi differenti

Mapping diretto

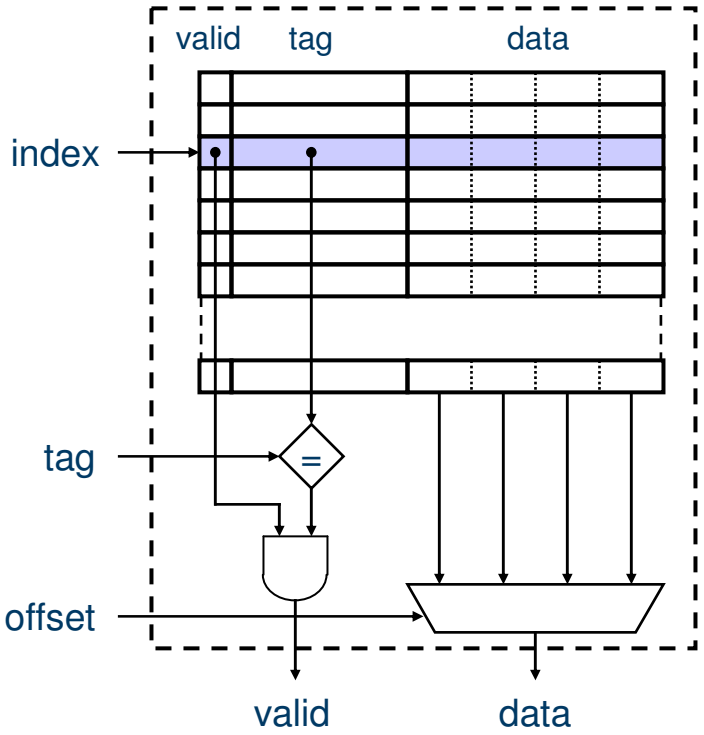
L'indirizzo in memoria centrale viene visto come costituito da tre campi di opportune dimensioni

- Index
  - Indirizzo della linea di cache
  - Il numero di bit di questo campo dipende dalle dimensioni della cache
- Tag
  - La parte alta dell'indirizzo
  - Viene confrontato con il tag salvato in cache all'indirizzo di index
  - Se i tag sono uguali si ha un cache hit
- Offset
  - Indica una particolare word, halfword o byte in una linea di cache

Inoltre è necessario disporre di un valid bit

- Indica se i dati presenti in una linea di cache sono validi

Direct mapping



Mapping completamente associativo

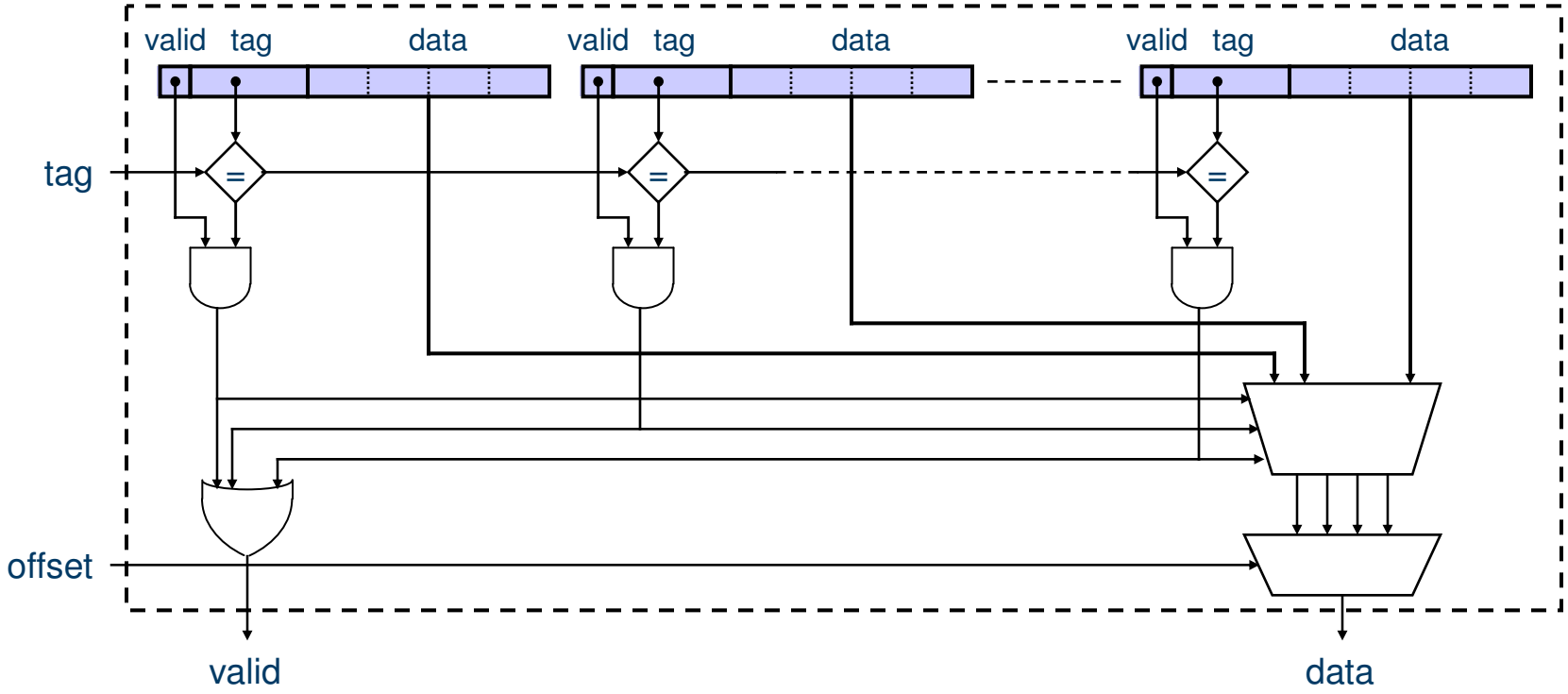
L'indirizzo in memoria centrale viene visto come costituito da due campi di opportune dimensioni

- Tag
  - La parte alta dell'indirizzo
  - Viene confrontato con tutti i tag salvati in cache simultaneamente
  - Se i tag sono uguali si ha un cache hit
- Offset
  - Indica una particolare word, halfword o byte in una linea di cache

Inoltre è necessario disporre di un valid bit

- Indica se i dati presenti in una linea di cache sono validi

Mapping completamente associativo



Mapping set-associativo

È un compromesso

- Tra il mapping diretto e quello completamente associativo

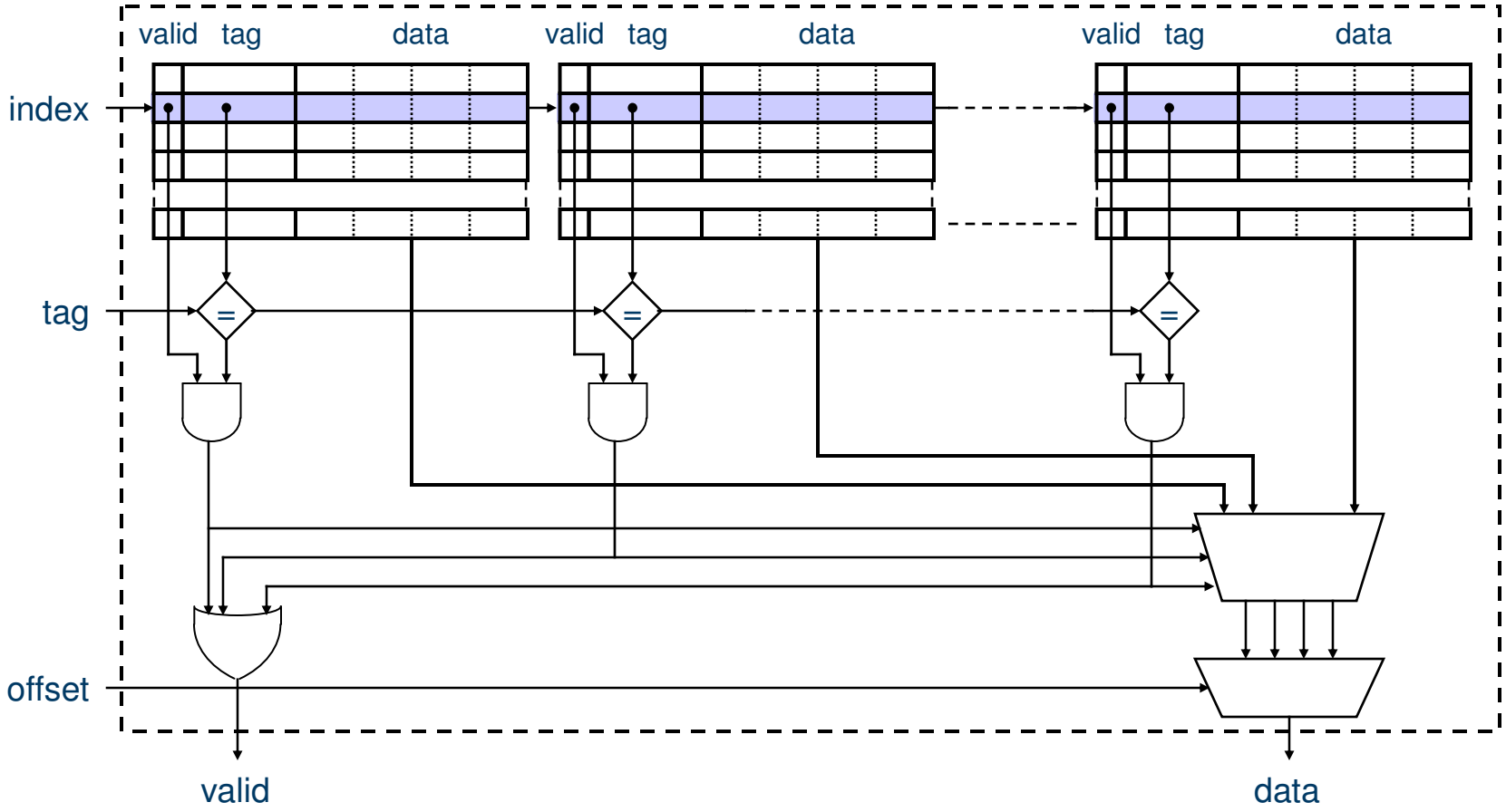
L'indirizzo della memoria centrale è scomposto in tre campi

- Index, Offset e Tag
  - Come nel caso di mapping diretto

Ma…

- Ad ogni indirizzo in cache corrispondono più indirizzi di memoria
  - 2, 4, ..., 32 set
- I tag dei diversi set corrispondenti allo stesso indice vengono confrontati con il tag richiesto simultaneamente
  - Come nel caso di mapping completamente associativo

Mapping set-associativo



Politiche di sostituzione

Sono tecniche per decidere quale blocco sosituire

- Quando una cache completamente associativa è piena
- Quando la linea di una cache set-associativa è piena
- Per le cache a mapping diretto non c'è scelta

Casuale

- Semplice e mediamente efficiente

LRU: Least-Recently Used

- Sostituire il blocco usato meno di recente

FIFO: First-In-First-Out

- Inserisce i blocchi in una coda al momento della copia in cache
- Legge dalla coda il blocco da sostituire

Politiche di scrittura

Quando un dato presente in cache viene aggiornato

- La memoria cache deve essere aggiornata
- La memoria centrale deve essere aggiornata

Write-through

- La memoria centrale e la cache sono scritte simultaneamente
  - Facile da implementare
  - Il processore deve rispettare le tempistiche della memoria centrale lenta
  - Possono verificarsi scritture superflue

Write-back

- La memoria centrale viene aggiornata solo quando in cache è necessario sotituire un blocco che è stato modificato
  - È necessario un bit aggiuntivo, detto dirty bit, che viene scritto ogniqualvolta si aggiorna il vontenuto di un blocco in cache
  - Riduce significativamente il numero di accessi alla memoria centrale lenta