

La gestione dell'I/O

Direct Memory Access I/O

Sandro Savino (sandro.savino@dei.unipd.it)

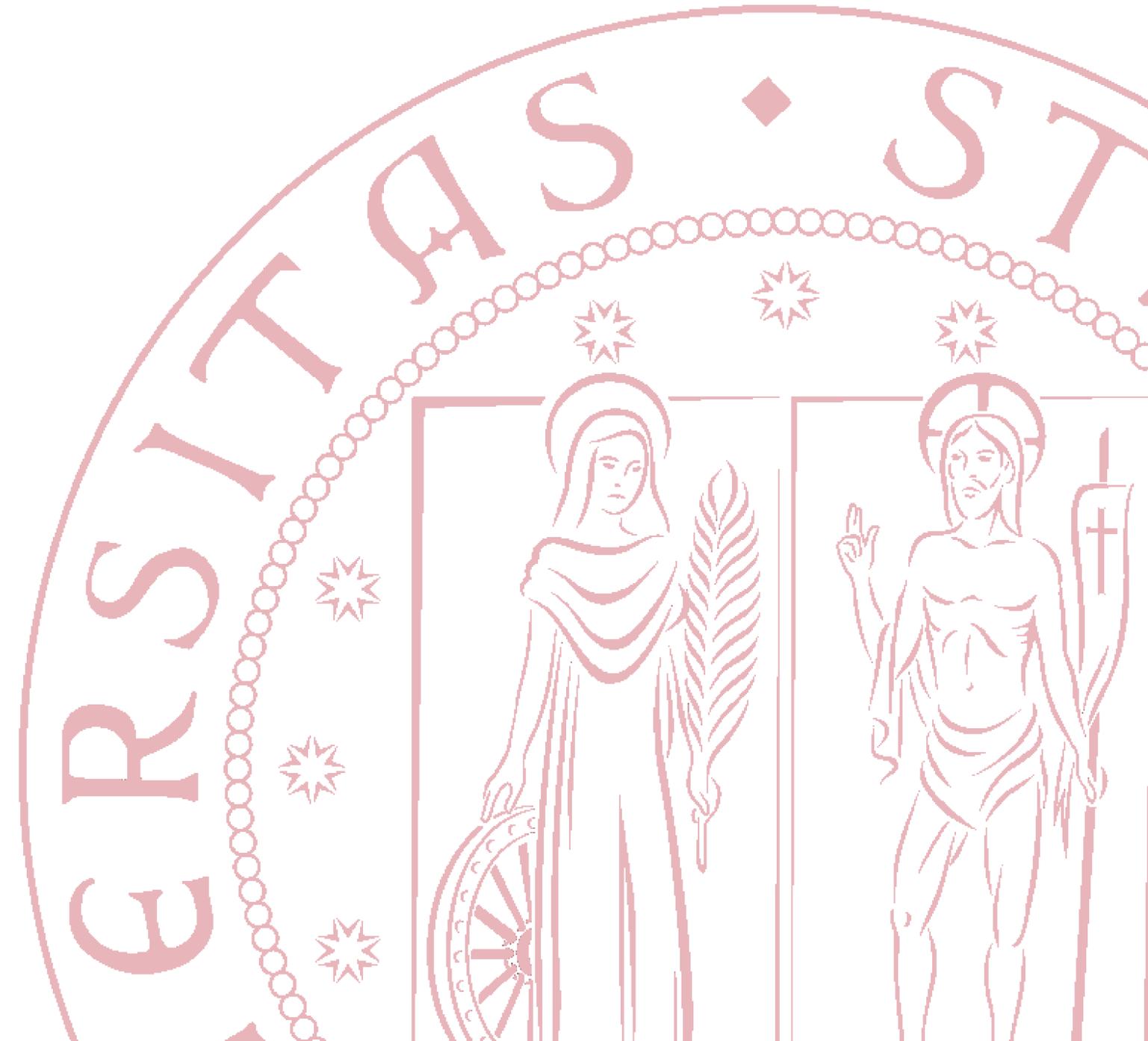
Department of Information Engineering, University of Padova

Argomenti:

- La gestione dell'I/O tramite DMA
- Arbitraggio del bus

Materiale:

- Capitolo 11



I/O Programmato: pro e contro

PRO

- Molto facile da realizzare
- Tempo di risposta molto veloce(*): appena il dato è pronto viene letto / inviato



(*) se il dispositivo è pronto ...

CONTRO

- Mentre è in attesa che il dispositivo sia pronto, il processore non può fare niente altro!
- L'inefficienza peggiora più alta è la differenza di velocità tra il processore e il dispositivo di I/O

I/O a Interruzioni: pro e contro

CONTRO

- Sistema più complesso
- È necessario software dedicato
- È necessario hardware dedicato



PRO

- Efficiente: il processore non deve stare in busy waiting
- Usando HW nell'identificazione e nella gestione della priorità, abbiamo tempi di risposta rapidissimi
 - Hardware poll
 - Vectored IRQ
 - Banked registers

Performance dell'I/O

- Parlando di gestione di I/O ci interessa
 - L'impatto sulle performance dell'elaboratore
 - L'efficienza, ovvero il tempo di risposta
- Parlando di I/O un altro parametro che ci interessa è il «**data rate**», ovvero quanti dati al secondo si possono trasferire
- In un sistema a interrupt in cui la gestione dell'I/O in totale richiede $10\mu\text{s}$ (tempo di esecuzione della RSI, salvataggio del contesto compreso), quanti dati ci permette di trasferire?

Data rate

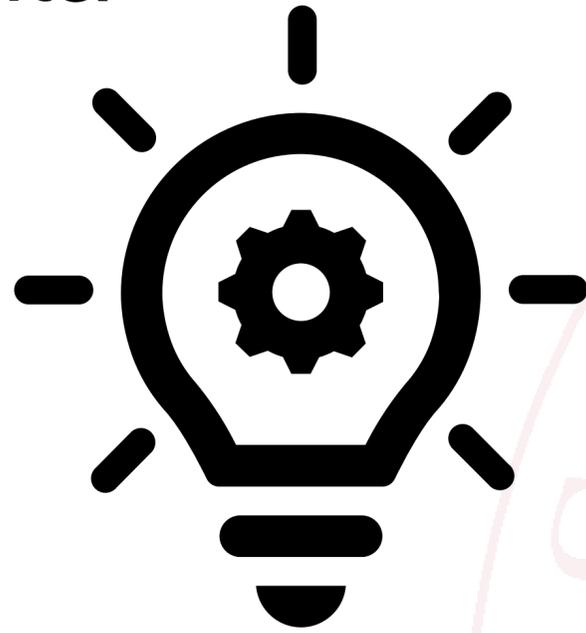
- Ad ogni esecuzione la RSI gestisce un tot limitato di dati
- Se una RSI impiega $10\mu\text{s}$ per essere eseguita, potrà essere eseguita al massimo 100 mila volte in un secondo
 - Se ad ogni esecuzione la RSI gestisce ad es. 16 bit, il limite massimo del data rate sarà di 200 KB/s

Nota: a questa velocità, il processore sarà **costantemente** impegnato a gestire gli IRQ e non potrà fare altro!

- Se il device produce più di 100 mila dati /secondo il processore non riuscirà a leggerli tutti e avremo una **perdita di dati** !
- Dimezzando la velocità, a 100 KB/s, il processore sarà comunque impegnato per il 50% del tempo!

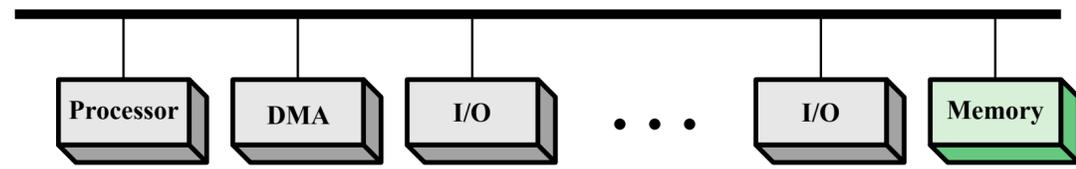
Una nuova idea!

- Per il trasferimento di dati da/verso la memoria ad alta velocità il sistema ad interrupt non va bene.
- Il problema è che il processore viene interrotto ogni volta che una piccola parte dei dati è pronta

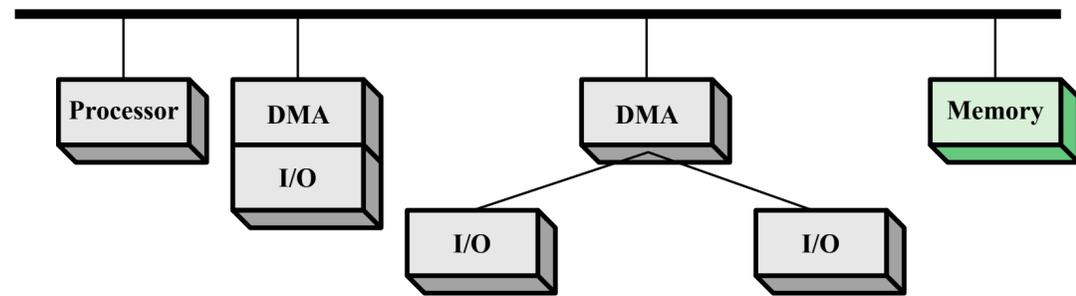


Ma se il compito del processore è solo quello di prelevare i dati dal dispositivo e metterli in memoria (o viceversa), questo semplice compito non potrebbe farlo qualcun altro?

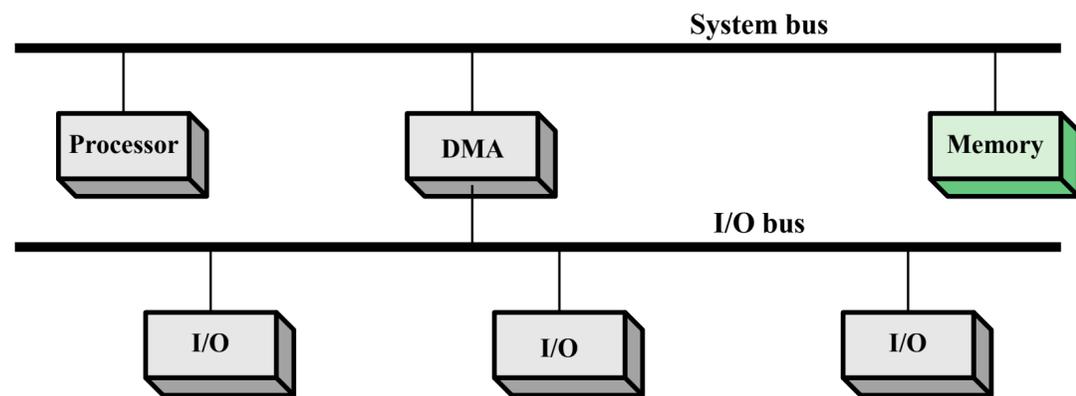
DMA – Direct Memory Access



(a) Single-bus, detached DMA



(b) Single-bus, Integrated DMA-I/O

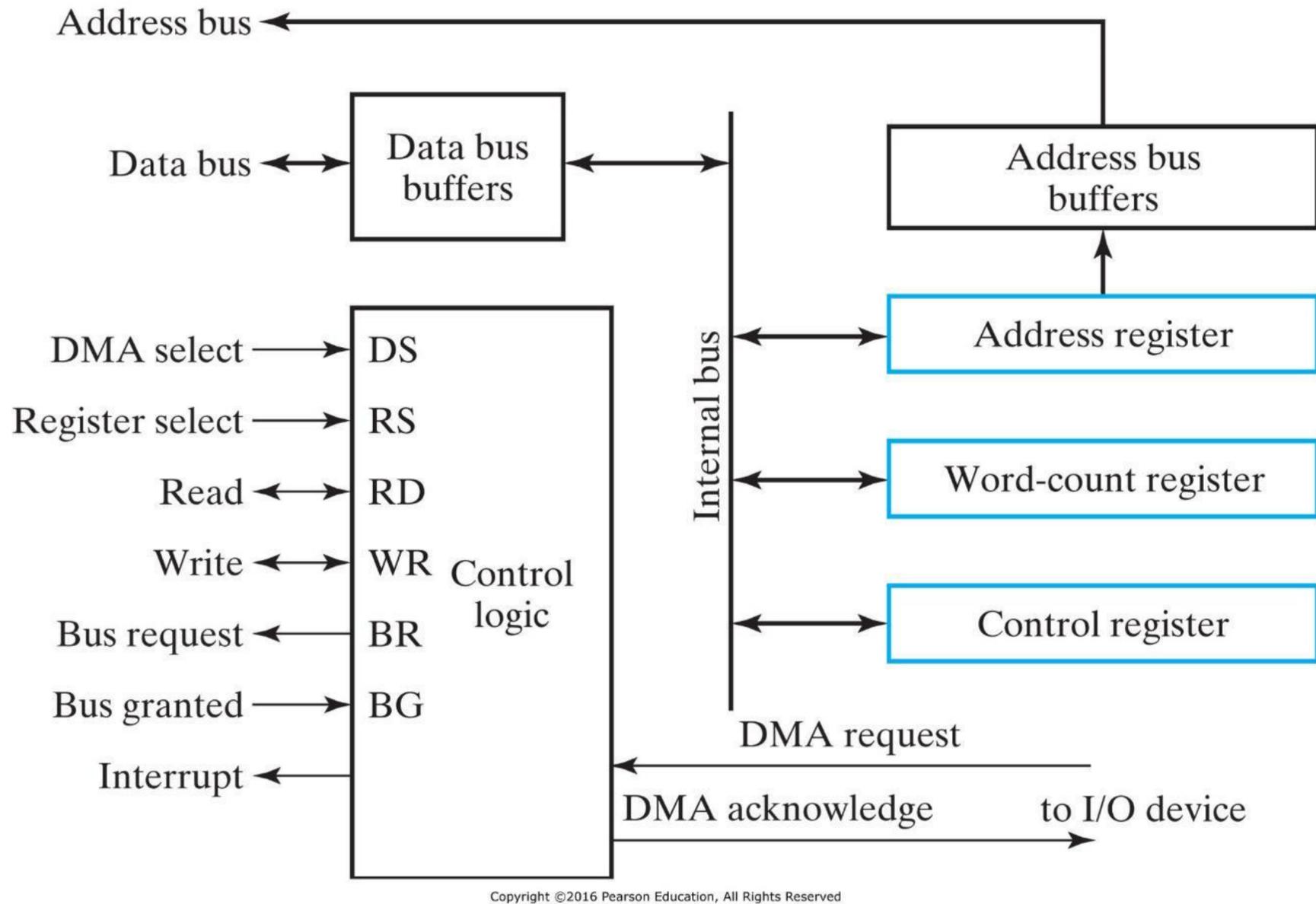


(c) I/O bus

Figure 7.14 Alternative DMA Configurations

- Aggiungiamo al nostro elaboratore un DMA Controller
- Il DMA ha accesso ai dispositivi di I/O e alla memoria tramite il bus
- Il DMA riceve le richieste di I/O da parte dei device che controlla e avverte il processore
- Il processore gli dice quanti dati vuole leggere, da che dispositivo e la posizione iniziale dove scriverli in memoria
- Il DMA si occupa del trasferimento e quando ha terminato avverte il processore con un IRQ

DMA Controller



- È come una interface unit: è in mezzo tra la CPU e il device di I/O
 - Con il device è collegato con due linee per hand-shaking
 - Con la CPU è collegato con alcune nuove linee: BR e BG per chiedere l'uso del bus

Al suo interno troviamo:

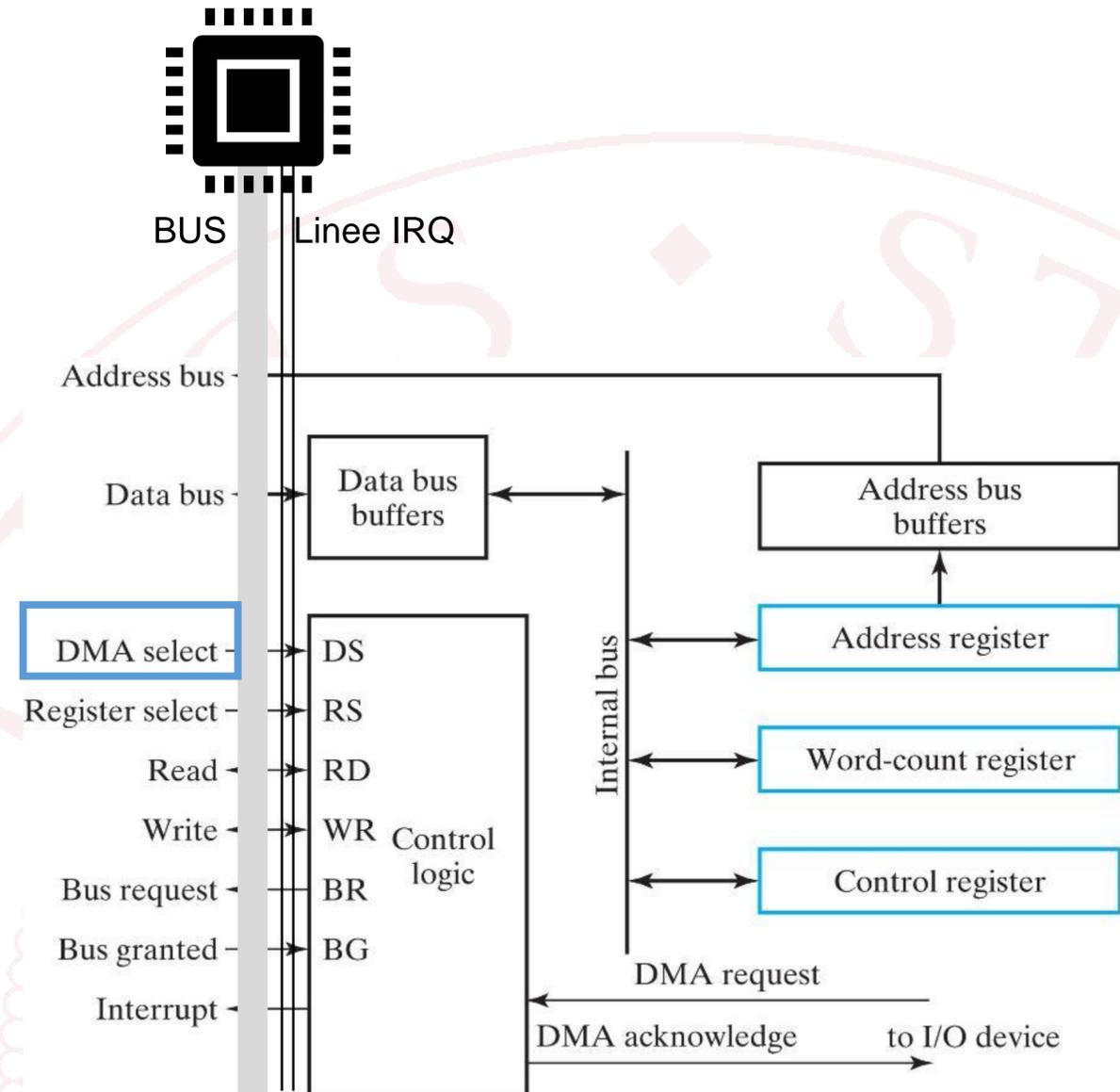
- Un registro per l'indirizzo di memoria
- Un registro per il numero di dati da trasferire
- Un registro control per impostare il transfer mode (R, W)

Il DMA controller è un piccolo processore

- Può comandare più device di I/O e gestirne la priorità

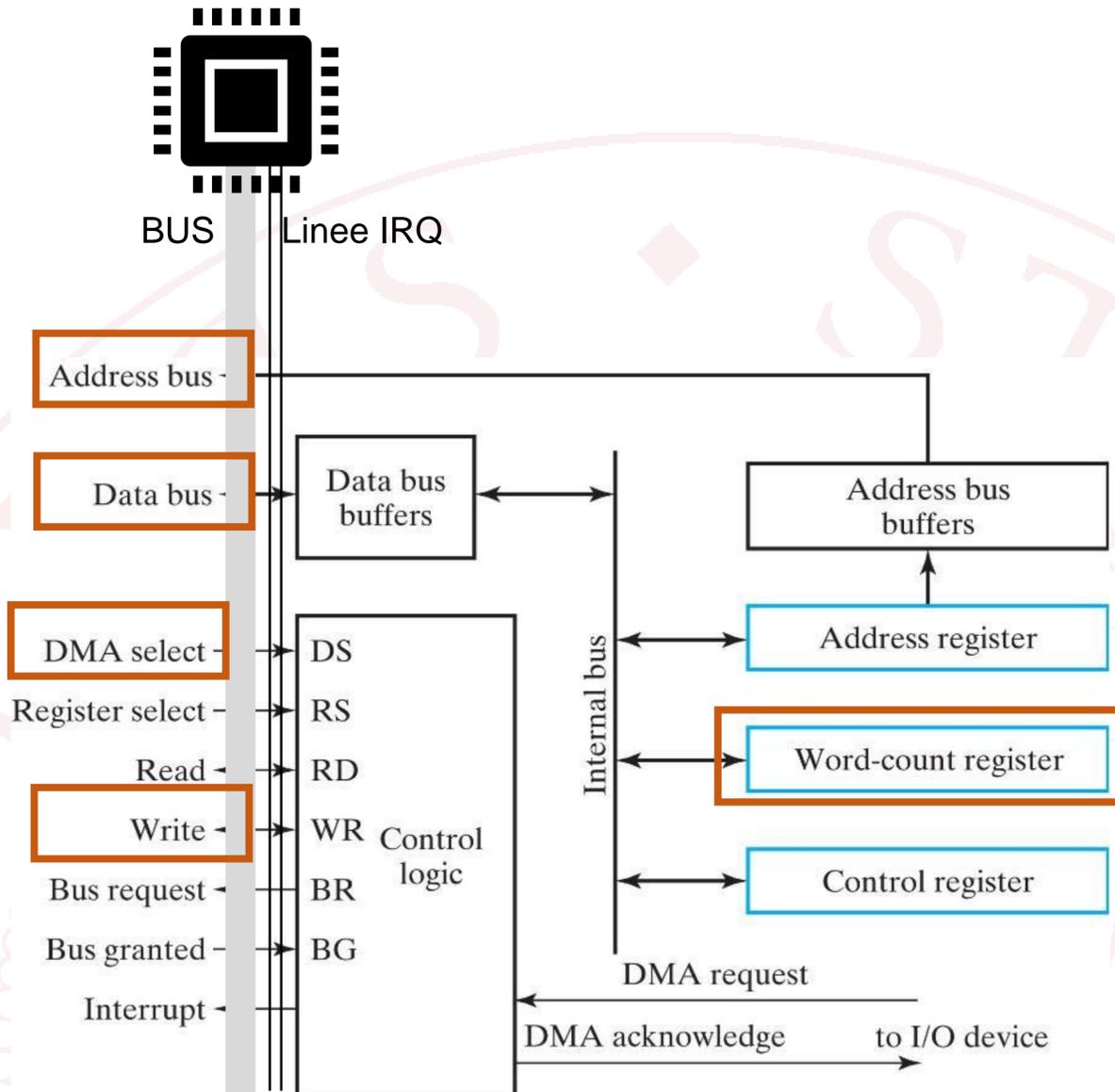
DMA in output

- Il processore indica
 - Device di I/O su cui scrivere



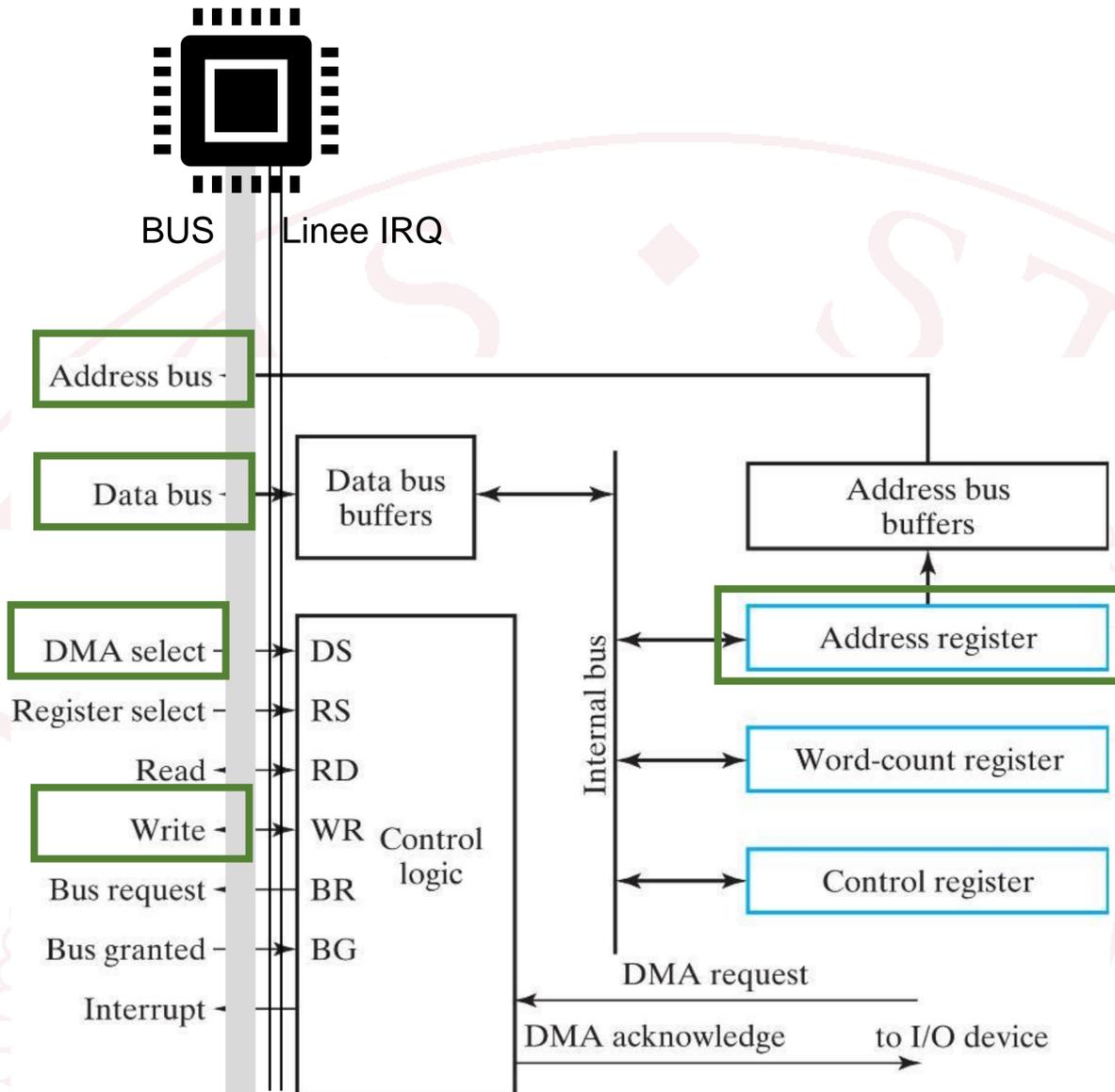
DMA in output

- Il processore indica
 - Device di I/O su cui scrivere
 - Quanti dati trasferire



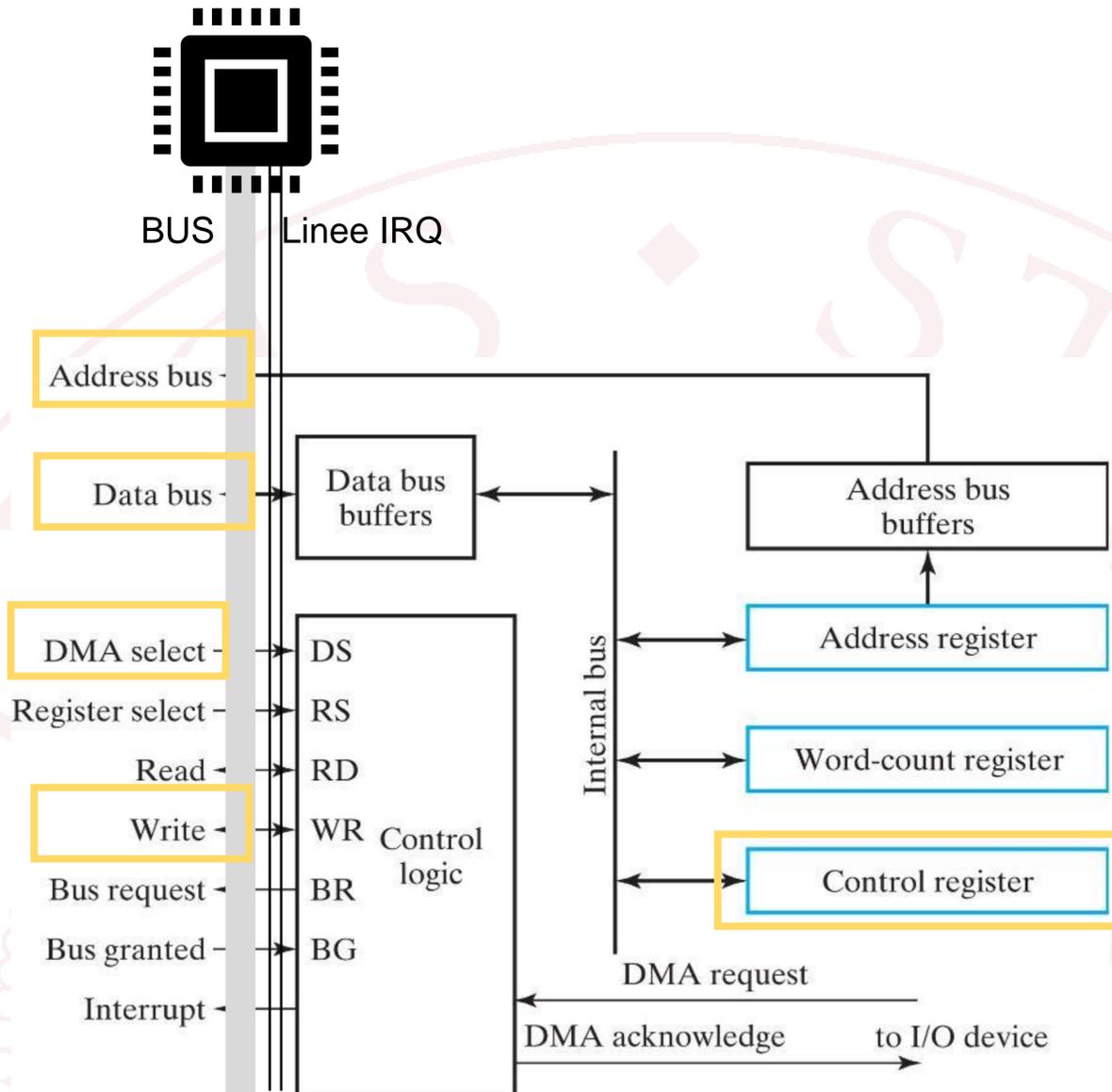
DMA in output

- Il processore indica
 - Device di I/O su cui scrivere
 - Quanti dati trasferire
 - Indirizzo di memoria da cui leggere



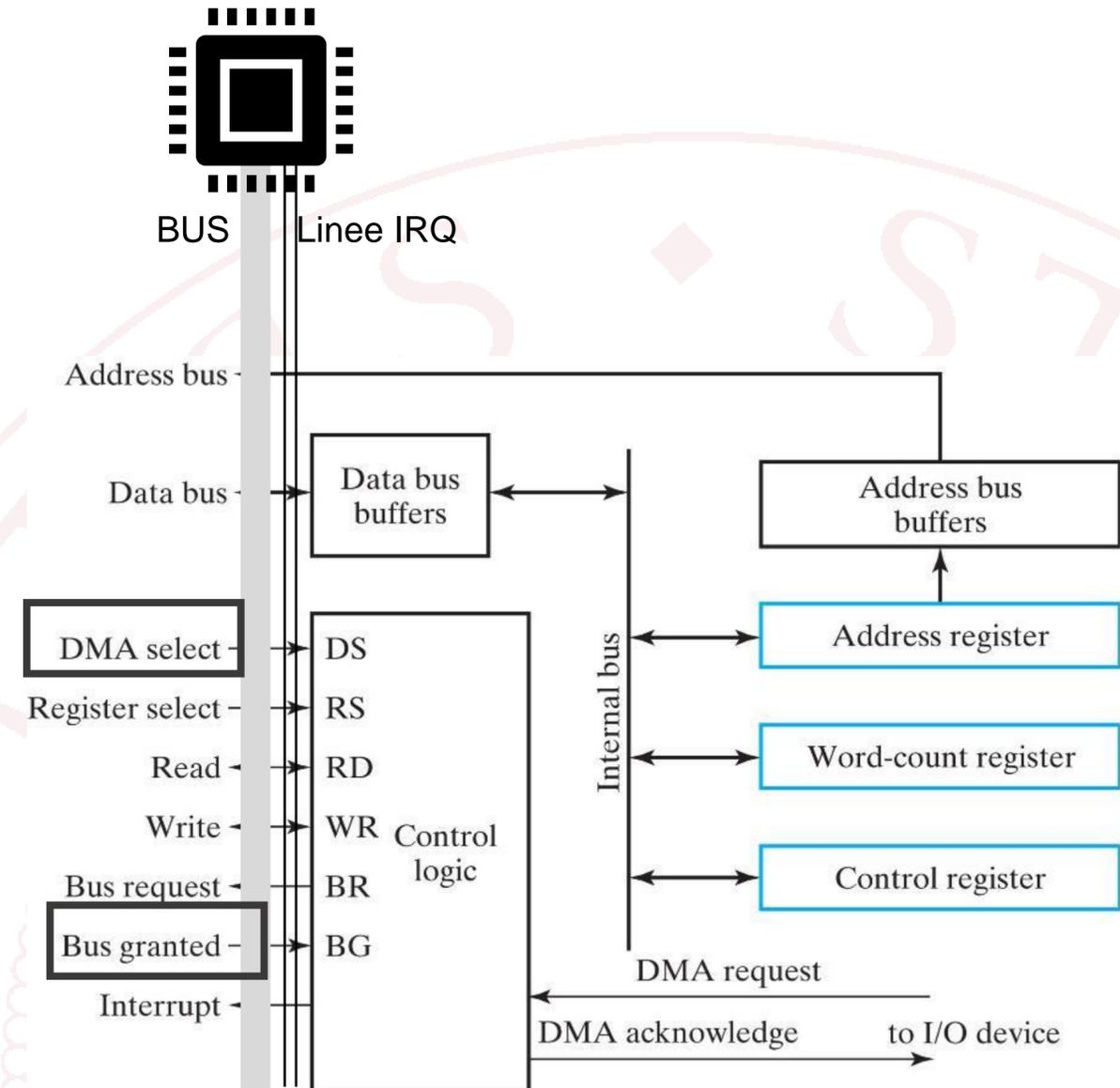
DMA in output

- Il processore indica
 - Device di I/O su cui scrivere
 - Quanti dati trasferire
 - Indirizzo di memoria da cui leggere
 - Un bit di controllo che dice di scrivere



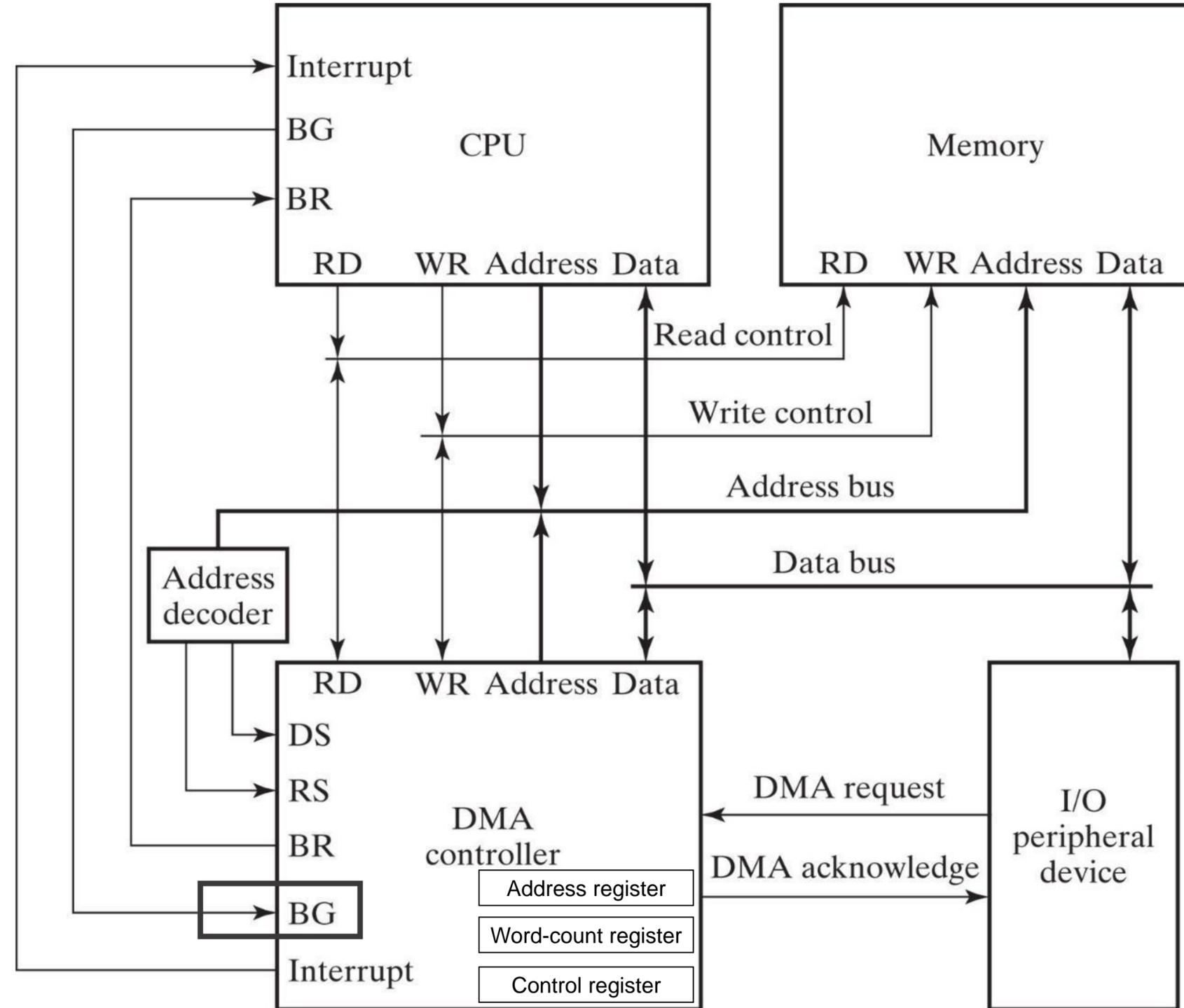
DMA in output

- Il processore indica
 - Device di I/O su cui scrivere
 - Quanti dati trasferire
 - Indirizzo di memoria da cui leggere
 - Un bit di controllo che dice di scrivere
 - Un comando di start



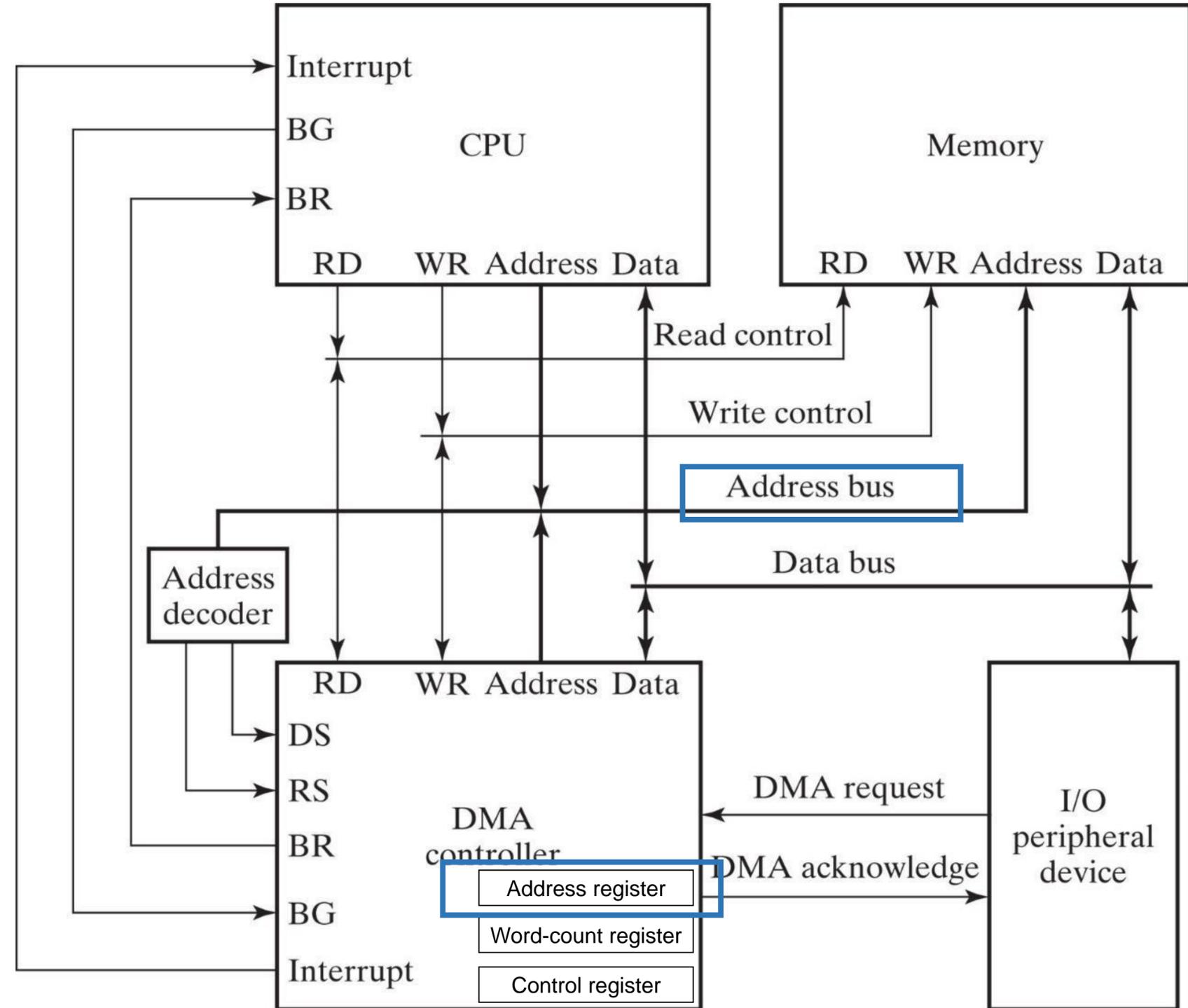
DMA in output

- Il DMA controller:
 - riceve il controllo del bus con il segnale BG



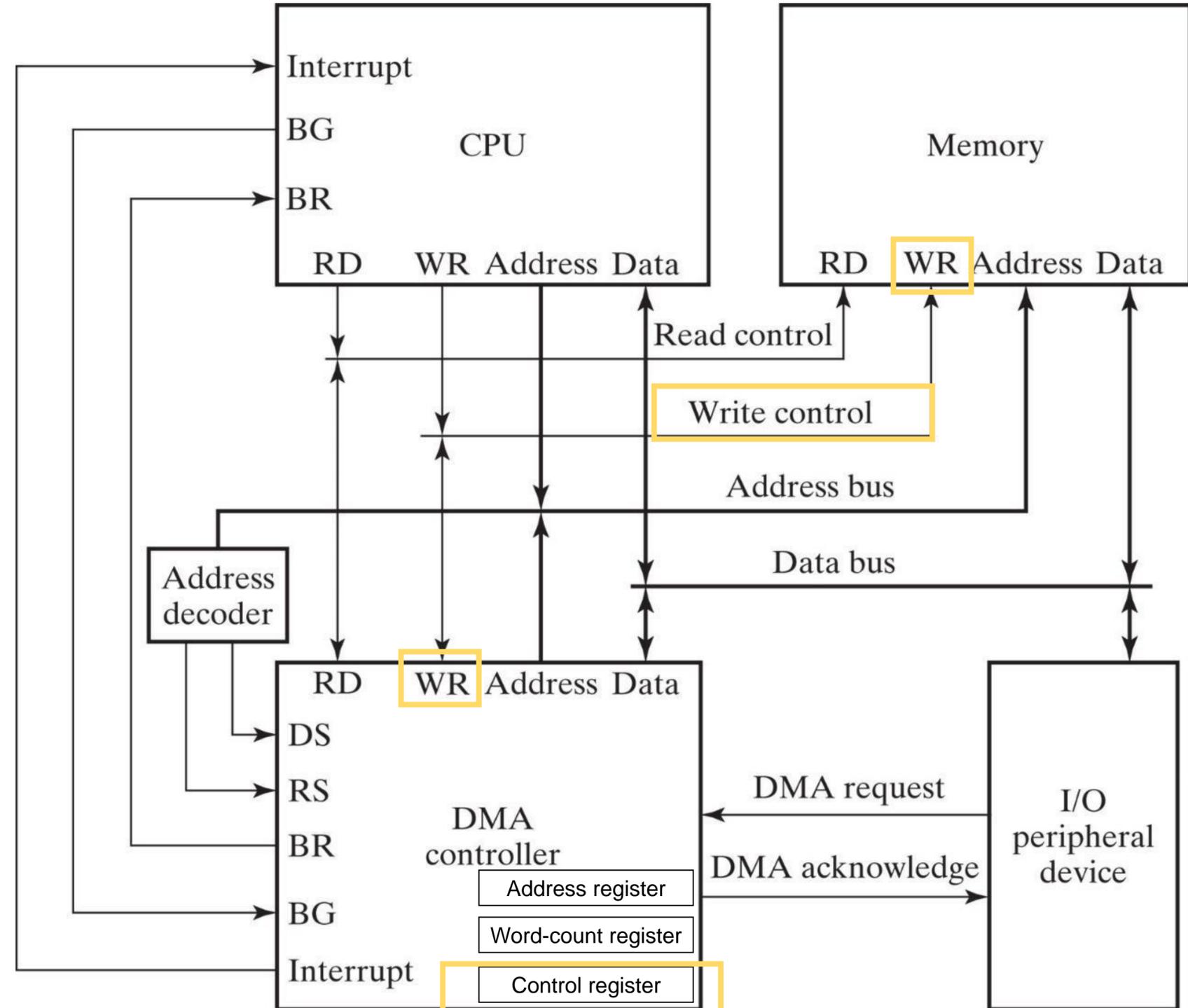
DMA in output

- Il DMA controller:
 - riceve il controllo del bus con il segnale BG
 - mette sull'address bus il valore nel suo address register



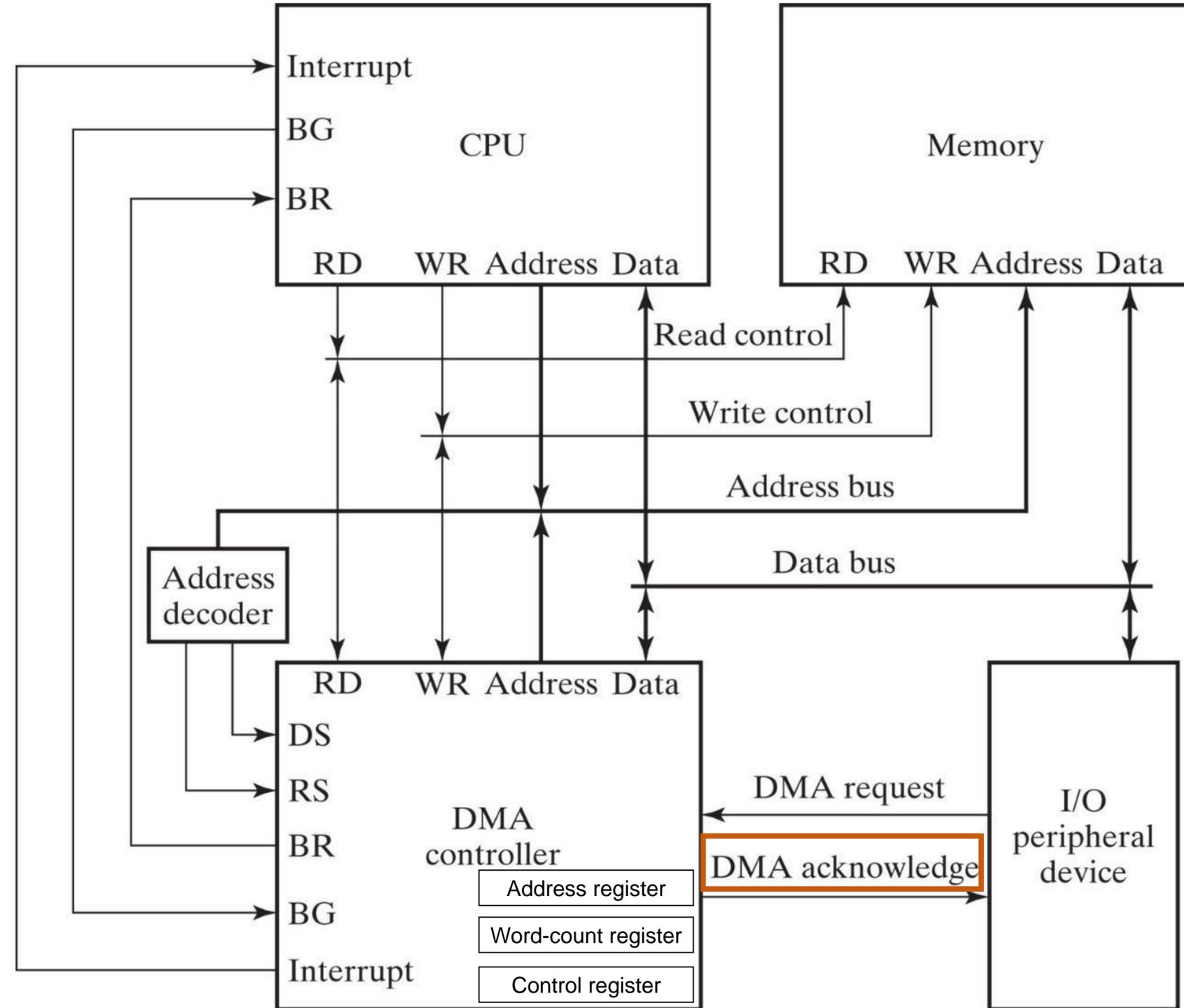
DMA in output

- Il DMA controller:
 - riceve il controllo del bus con il segnale BG
 - mette sull'address bus il valore nel suo address register
 - attiva la linea WR a seconda di quanto scritto nel control register



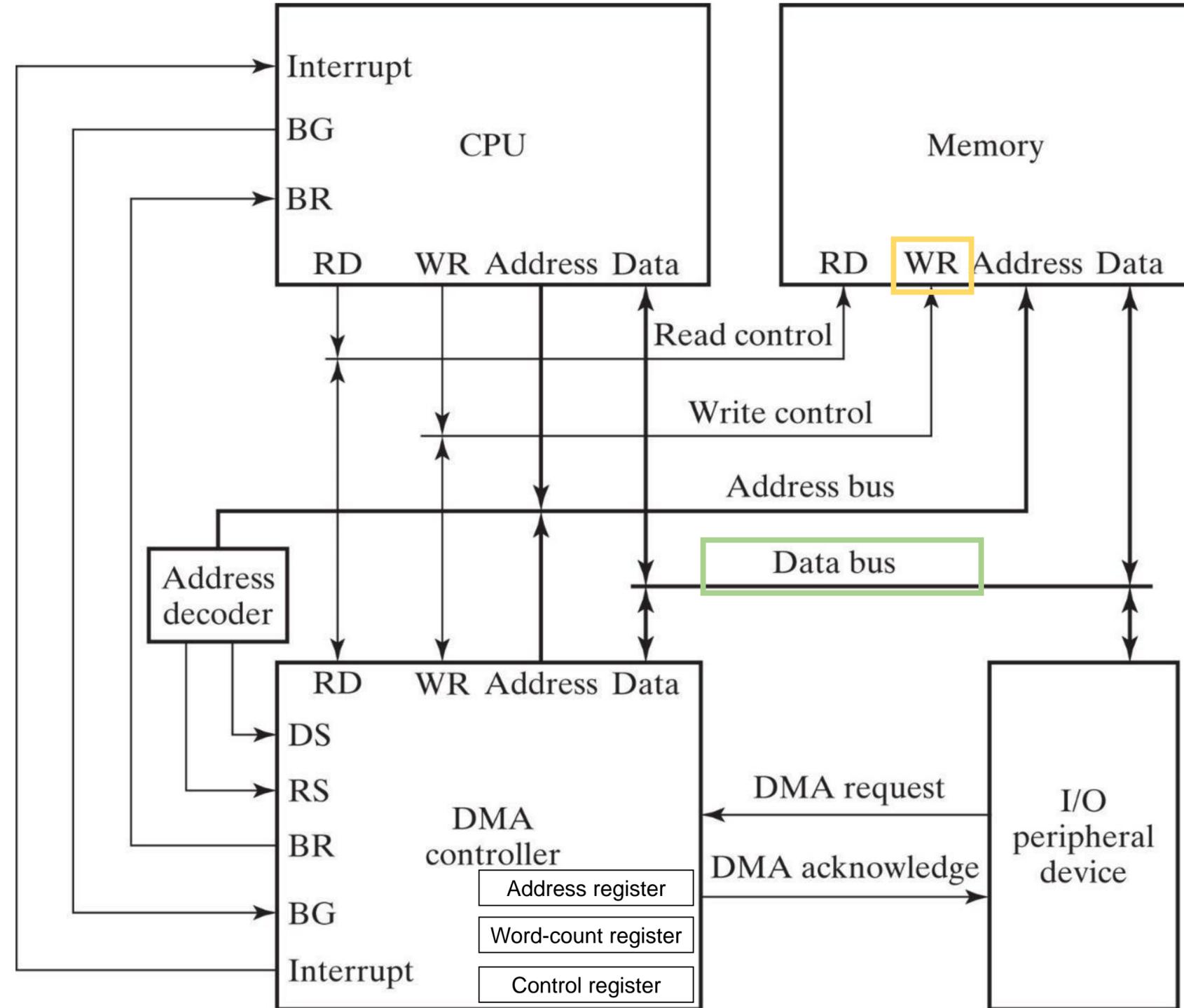
DMA in output

- Il DMA controller:
 - riceve il controllo del bus con il segnale BG
 - mette sull'address bus il valore nel suo address register
 - attiva la linea WR a seconda di quanto scritto nel control register
 - invia un segnale DMA acknowledge al device di I/O che aveva fatto la richiesta di DMA



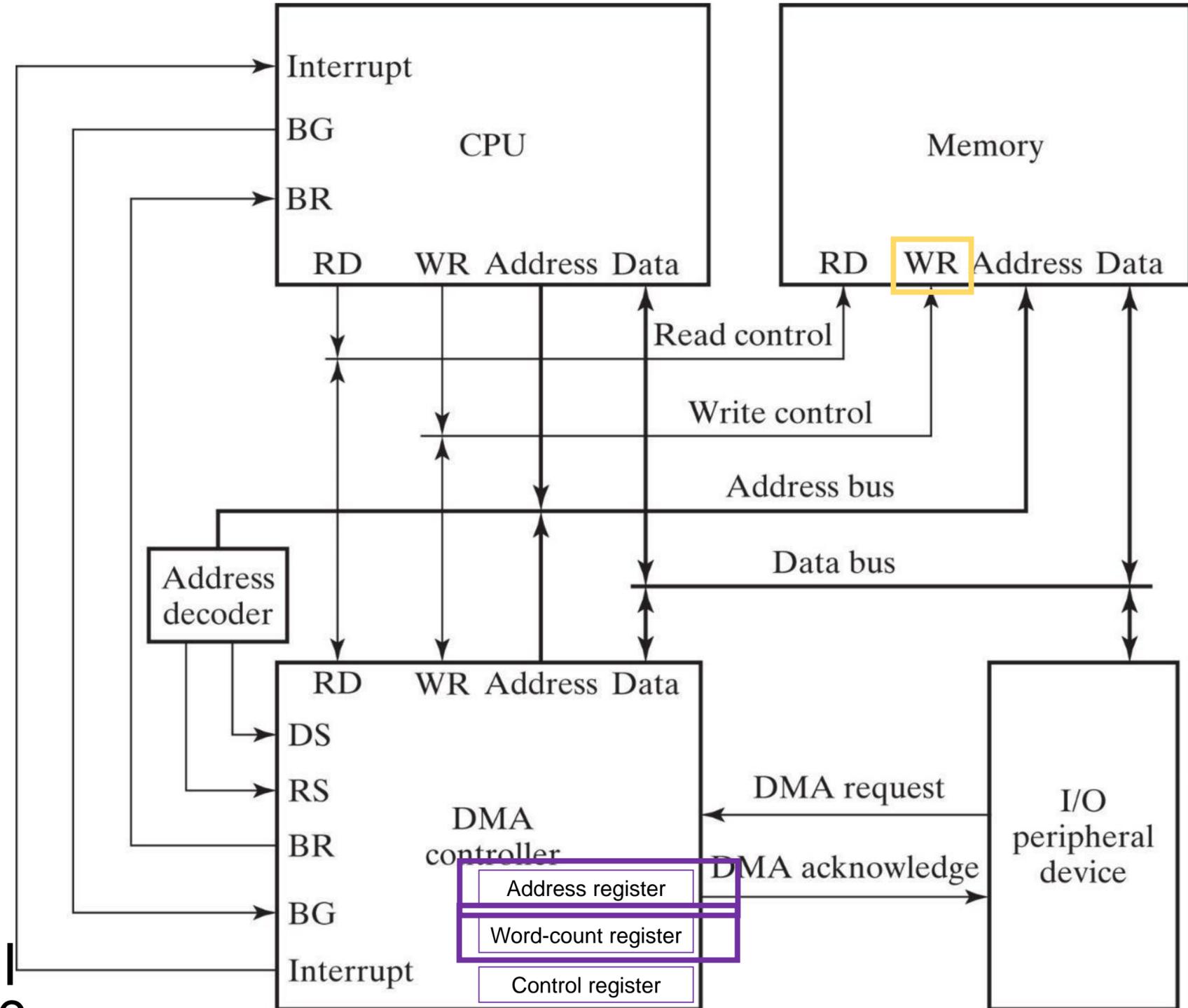
DMA in output

- Il DMA controller:
 - riceve il controllo del bus con il segnale BG
 - mette sull'address bus il valore nel suo address register
 - attiva la linea WR a seconda di quanto scritto nel control register
 - invia un segnale DMA acknowledge al device di I/O che aveva fatto la richiesta di DMA
- Il DEVICE:
 - mette un dato sul bus dei dati, che viene quindi scritto in memoria



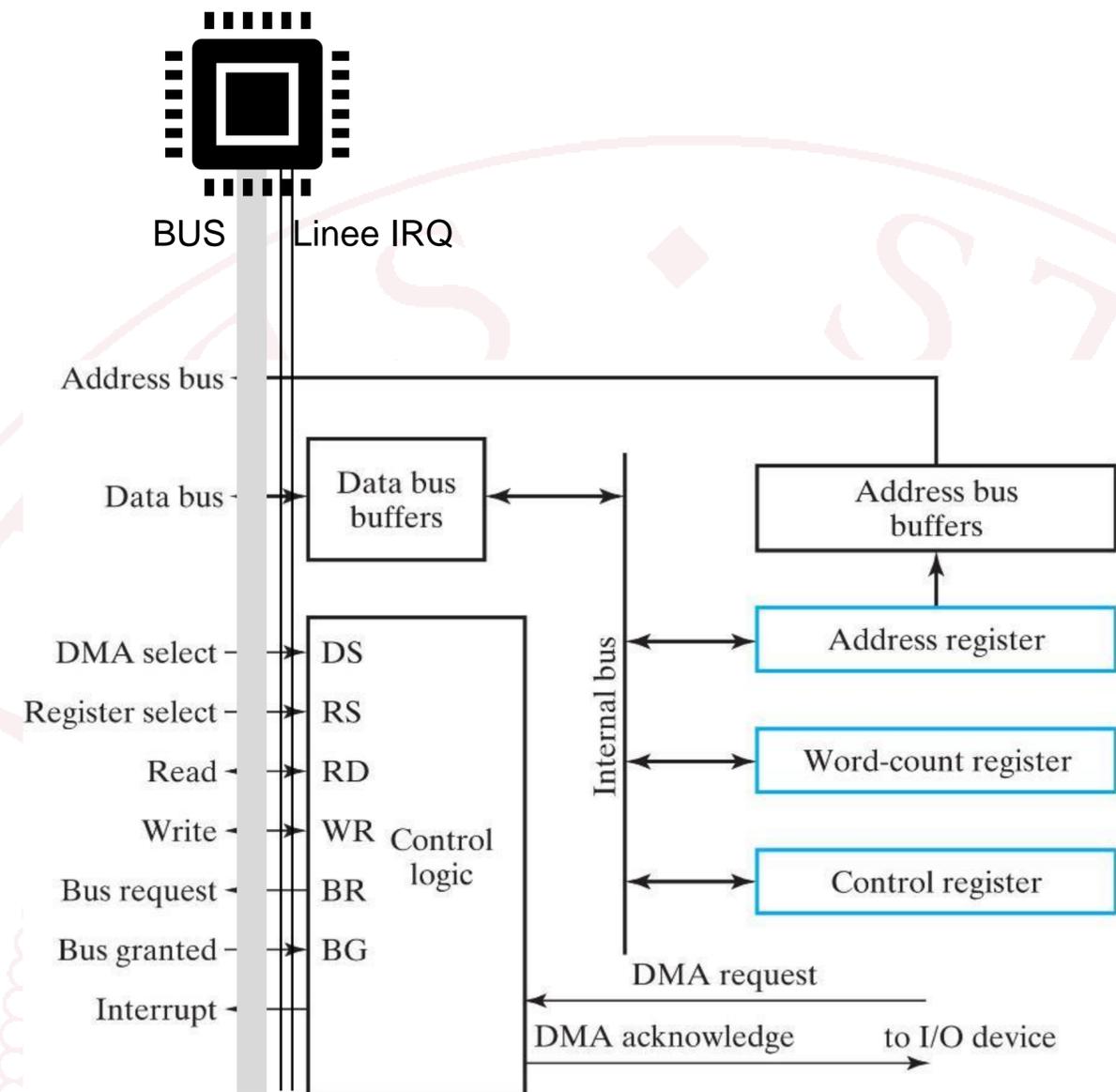
DMA in output

- Il DMA controller:
 - riceve il controllo del bus con il segnale BG
 - mette sull'address bus il valore nel suo address register
 - attiva la linea WR a seconda di quanto scritto nel control register
 - invia un segnale DMA acknowledge al device di I/O che aveva fatto la richiesta di DMA
- Il DEVICE:
 - mette un dato sul bus dei dati, che viene quindi scritto in memoria
- Il DMA controller:
 - incrementa l'indirizzo e decrementa il word-count avvertendo la CPU se è 0



DMA in output (input)

- Il processore indica
 - Device di I/O su cui (leggere)
 - Quanti dati trasferire
 - Indirizzo di memoria da cui (scrivere)
 - Un bit di controllo che dice di (scrivere)
 - Da il comando START
- Il controller
 - È in busy waiting sul dispositivo
 - Trasferisce il dato appena pronto
 - Decrementa il contatore dei dati e aggiorna il puntatore a memoria
 - Quando il contatore è zero, avverte il processore con un IRQ



Arbitraggio del bus

- Per trasferire i dati, il DMA deve passare per il BUS, che è condiviso con il processore
- Può capitare che il bus serva contemporaneamente sia alla CPU che al DMA:
- in questo caso tipicamente si dà precedenza al DMA (la perdita di dati è considerata più grave):
 - Il DMA procede a trasferire i dati
 - Il processore dovrà attendere per accedere alla memoria



Arbitraggio del bus: cycle stealing

Il processore accede alla memoria per il fetch: se il fetch non si può fare, il processore non ha nulla su cui lavorare!

→ Quando il bus è occupato dal DMA, il processore è di fatto fermo: Il processore perde un ciclo «cycle stealing»

- Per evitare che il processore resti bloccato troppo a lungo, il trasferimento non viene fatto tutto di un colpo (burst) ma viene fatto un dato alla volta: l'arbitraggio del bus è fatto ad ogni ciclo di memoria

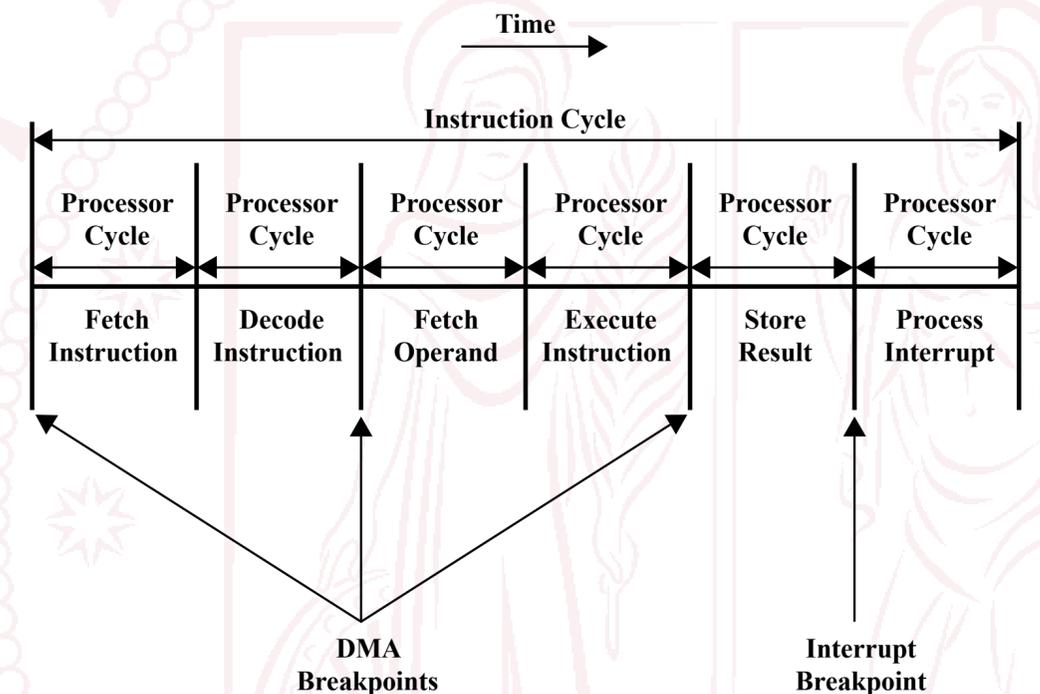


Figure 7.13 DMA and Interrupt Breakpoints During an Instruction Cycle

I/O con DMA: pro e contro

CONTRO

- Sistema più complesso
- È necessario software dedicato
- È necessario hardware dedicato
- Impegna il bus di sistema



PRO

- Permette il trasferimento ad alta velocità di dati dai dispositivi di I/O alla memoria (e viceversa)
- Il processore è impegnato solo quando il trasferimento è stato completato

Sistemi multiprocessore

- Abbiamo visto una tecnica con cui condividere il bus (e la memoria) tra due dispositivi: il DMA controller e la CPU
- Con la stessa tecnica possiamo pensare di condividere queste risorse con altri dispositivi, da esempio dei processori supplementari:
 - Processori ausiliari (es. specializzati per la grafica)
 - Processori paritari multipli (multiprocessore), che permettono il calcolo parallelo e la ridondanza

