

 POLITECNICO DI MILANO

Dipartimento di
Elettronica e Informazione

Memorie, processi e temi d'esame

Matteo Ferroni
matteo.ferroni@polimi.it

26/01/2016



POLITECNICO
DI MILANO



Agenda

- (20') La gerarchia di memoria (teoria)
- (10') Cache e tempi di accesso (Es.3 - TE 01/09/2015)
- (20') Memoria fisica e memoria virtuale (Es.3 - TE ?)
- (15') Stato dei processi (Es.4 - TE 19/02/2015)
- (30') Ordine superiore e ricorsione (Es3 - TE 19/2/2015)
- (20') Iterazione vs. Ricorsione (Es.3 - TE 19/02/2015)
- (20') Conggettura di Goldbach (Es.1 - TE 19/02/2015)

La memoria cache

Il problema della memoria: costo vs. prestazioni

La memoria cache

Il problema della memoria: costo vs. prestazioni

Obiettivo:

La memoria cache

Il problema della memoria: costo vs. prestazioni

Obiettivo:

- fornire agli **utenti** una memoria grande e veloce

La memoria cache

Il problema della memoria: costo vs. prestazioni

Obiettivo:

- fornire agli **utenti** una memoria grande e veloce
- fornire al **processore** i dati alla velocità con cui è in grado di elaborarli

La memoria cache

Il problema della memoria: costo vs. prestazioni

Obiettivo:

- fornire agli **utenti** una memoria grande e veloce
- fornire al **processore** i dati alla velocità con cui è in grado di elaborarli

Problema: Il tasso di crescita nella velocità dei processori non è stato seguito da quello delle memorie

La memoria cache

Il problema della memoria: costo vs. prestazioni

Obiettivo:

- fornire agli **utenti** una memoria grande e veloce
- fornire al **processore** i dati alla velocità con cui è in grado di elaborarli

Problema: Il tasso di crescita nella velocità dei processori non è stato seguito da quello delle memorie

- Tempo di accesso alle **SRAM** (Static Random Access Memory):
 - 2 - 25ns al costo di \$100 - \$250 per Mbyte.

La memoria cache

Il problema della memoria: costo vs. prestazioni

Obiettivo:

- fornire agli **utenti** una memoria grande e veloce
- fornire al **processore** i dati alla velocità con cui è in grado di elaborarli

Problema: Il tasso di crescita nella velocità dei processori non è stato seguito da quello delle memorie

- Tempo di accesso alle **SRAM** (Static Random Access Memory):
 - 2 - 25ns al costo di \$100 - \$250 per Mbyte.
- Tempo di accesso alle **DRAM** (Dynamic Random Access Memory):
 - 60 - 120ns al costo di \$5 - \$10 per Mbyte.

La memoria cache

Il problema della memoria: costo vs. prestazioni

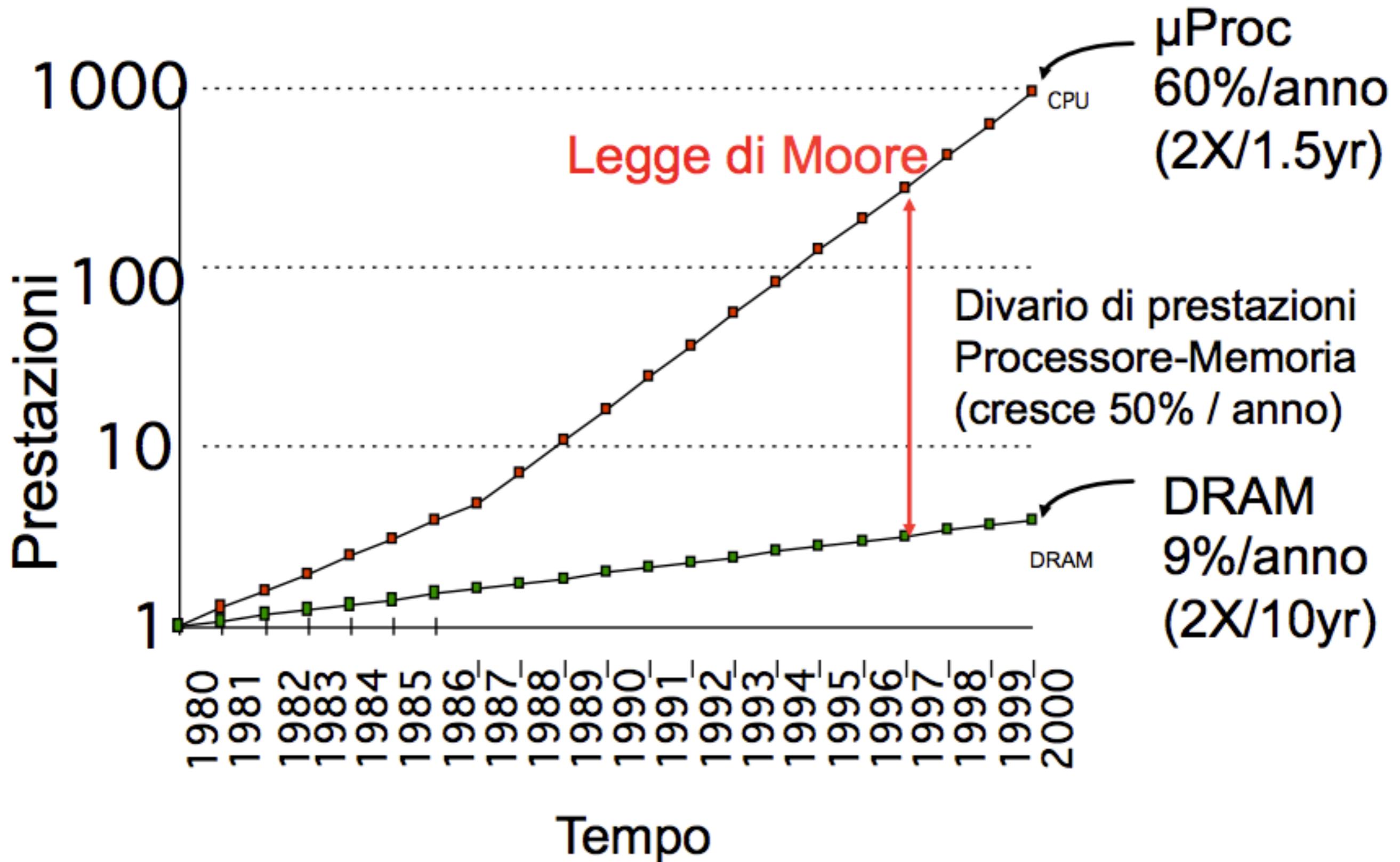
Obiettivo:

- fornire agli **utenti** una memoria grande e veloce
- fornire al **processore** i dati alla velocità con cui è in grado di elaborarli

Problema: Il tasso di crescita nella velocità dei processori non è stato seguito da quello delle memorie

- Tempo di accesso alle **SRAM** (Static Random Access Memory):
 - 2 - 25ns al costo di \$100 - \$250 per Mbyte.
- Tempo di accesso alle **DRAM** (Dynamic Random Access Memory):
 - 60 - 120ns al costo di \$5 - \$10 per Mbyte.
- Tempo di accesso al **disco**:
 - 10 - 20 million ns al costo di \$0.10 - \$0.20 per Mbyte.

Prestazioni di processori e cache



Gerarchia di memoria

Gerarchia di memoria

Tipicamente costituita da:

Gerarchia di memoria

Tipicamente costituita da:

1. **registri** contenuti nella CPU (qualche KB)

Gerarchia di memoria

Tipicamente costituita da:

1. **registri** contenuti nella CPU (qualche KB)
2. **cache** (da circa 32KB a circa 4096KB)

Gerarchia di memoria

Tipicamente costituita da:

1. **registri** contenuti nella CPU (qualche KB)
2. **cache** (da circa 32KB a circa 4096KB)
3. **memoria principale** (da circa 512MB a qualche GB)

Gerarchia di memoria

Tipicamente costituita da:

1. **registri** contenuti nella CPU (qualche KB)
2. **cache** (da circa 32KB a circa 4096KB)
3. **memoria principale** (da circa 512MB a qualche GB)
4. **dischi fissi** (da qualche centinaio di GB a qualche TB)

Gerarchia di memoria

Tipicamente costituita da:

1. **registri** contenuti nella CPU (qualche KB)
2. **cache** (da circa 32KB a circa 4096KB)
3. **memoria principale** (da circa 512MB a qualche GB)
4. **dischi fissi** (da qualche centinaio di GB a qualche TB)
5. **nastri magnetici** e **dischi ottici** (da qualche centinaio di GB a qualche TB per ogni supporto)

Gerarchia di memoria

Tipicamente costituita da:

1. **registri** contenuti nella CPU (qualche KB)
2. **cache** (da circa 32KB a circa 4096KB)
3. **memoria principale** (da circa 512MB a qualche GB)
4. **dischi fissi** (da qualche centinaio di GB a qualche TB)
5. **nastri magnetici** e **dischi ottici** (da qualche centinaio di GB a qualche TB per ogni supporto)

Man mano che ci si sposta verso il basso nella gerarchia aumenta il valore dei parametri fondamentali:

Gerarchia di memoria

Tipicamente costituita da:

1. **registri** contenuti nella CPU (qualche KB)
2. **cache** (da circa 32KB a circa 4096KB)
3. **memoria principale** (da circa 512MB a qualche GB)
4. **dischi fissi** (da qualche centinaio di GB a qualche TB)
5. **nastri magnetici** e **dischi ottici** (da qualche centinaio di GB a qualche TB per ogni supporto)

Man mano che ci si sposta verso il basso nella gerarchia aumenta il valore dei parametri fondamentali:

- aumenta il **tempo** di accesso;

Gerarchia di memoria

Tipicamente costituita da:

1. **registri** contenuti nella CPU (qualche KB)
2. **cache** (da circa 32KB a circa 4096KB)
3. **memoria principale** (da circa 512MB a qualche GB)
4. **dischi fissi** (da qualche centinaio di GB a qualche TB)
5. **nastri magnetici** e **dischi ottici** (da qualche centinaio di GB a qualche TB per ogni supporto)

Man mano che ci si sposta verso il basso nella gerarchia aumenta il valore dei parametri fondamentali:

- aumenta il **tempo** di accesso;
- aumenta la **capacità** di memorizzazione;

Gerarchia di memoria

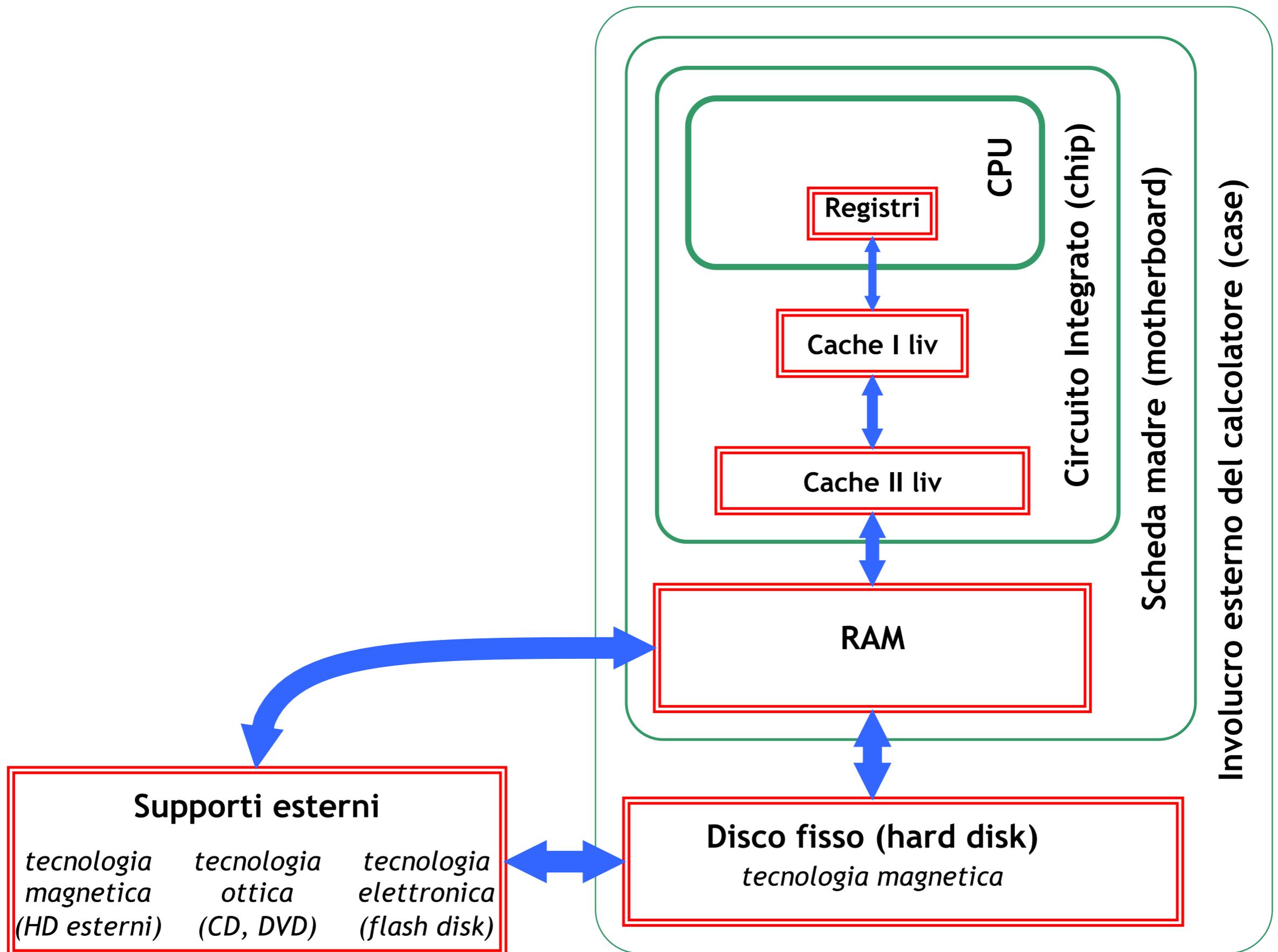
Tipicamente costituita da:

1. **registri** contenuti nella CPU (qualche KB)
2. **cache** (da circa 32KB a circa 4096KB)
3. **memoria principale** (da circa 512MB a qualche GB)
4. **dischi fissi** (da qualche centinaio di GB a qualche TB)
5. **nastri magnetici** e **dischi ottici** (da qualche centinaio di GB a qualche TB per ogni supporto)

Man mano che ci si sposta verso il basso nella gerarchia aumenta il valore dei parametri fondamentali:

- aumenta il **tempo** di accesso;
- aumenta la **capacità** di memorizzazione;
- ma diminuisce il **costo** per bit.

Una gerarchia di memoria



Caratteristiche dei diversi livelli

Livello	Capacità	Tempo di accesso	Transfer rate (GB/s)
Registri	~ 1 KB	~ 0.2 ns (1 ciclo di clock)	-
Cache I livello	~ 32 KB	~ 0.4 ns (2/4 cicli di clock)	-
Cache II livello	~ 1/2 MB	~ 1/2 ns (5/10 cicli di clock)	~ 100
Cache III livello	~ 2/8 MB	~ 5 ns	~ 50
Memoria centrale	~ 2/8 GB	~ 50 ns (1 ^a parola richiesta) ~ 10 ns (parole successive)	~ 5/10
Dischi interni	> 300 GB	~ 10 ms	0.15/0.6
Dischi esterni	> 300 GB	~ 10 ms	~ 0.05

Località

Località

- E' la proprietà fondamentale dei programmi che rende possibile sfruttare l'organizzazione gerarchica della memoria per incrementarne le prestazioni

Località

- E' la proprietà fondamentale dei programmi che rende possibile sfruttare l'organizzazione gerarchica della memoria per incrementarne le prestazioni
- **Località:** in ogni istante di tempo un programma accede a una parte relativamente piccola del suo spazio di indirizzamento

Località

- E' la proprietà fondamentale dei programmi che rende possibile sfruttare l'organizzazione gerarchica della memoria per incrementarne le prestazioni
- **Località**: in ogni istante di tempo un programma accede a una parte relativamente piccola del suo spazio di indirizzamento
 - Località **temporale**: se un dato viene referenziato in un dato istante, è probabile che lo stesso dato venga nuovamente richiesto entro breve

Località

- E' la proprietà fondamentale dei programmi che rende possibile sfruttare l'organizzazione gerarchica della memoria per incrementarne le prestazioni
- **Località**: in ogni istante di tempo un programma accede a una parte relativamente piccola del suo spazio di indirizzamento
 - Località **temporale**: se un dato viene referenziato in un dato istante, è probabile che lo stesso dato venga nuovamente richiesto entro breve
 - Località **spaziale**: Se un dato viene utilizzato in un dato istante, è probabile che dati posizionati in celle di memoria adiacenti vengano richiesti entro breve

Cache

Cache

- Memoria al livello superiore della gerarchia, veloce ma piccola

Cache

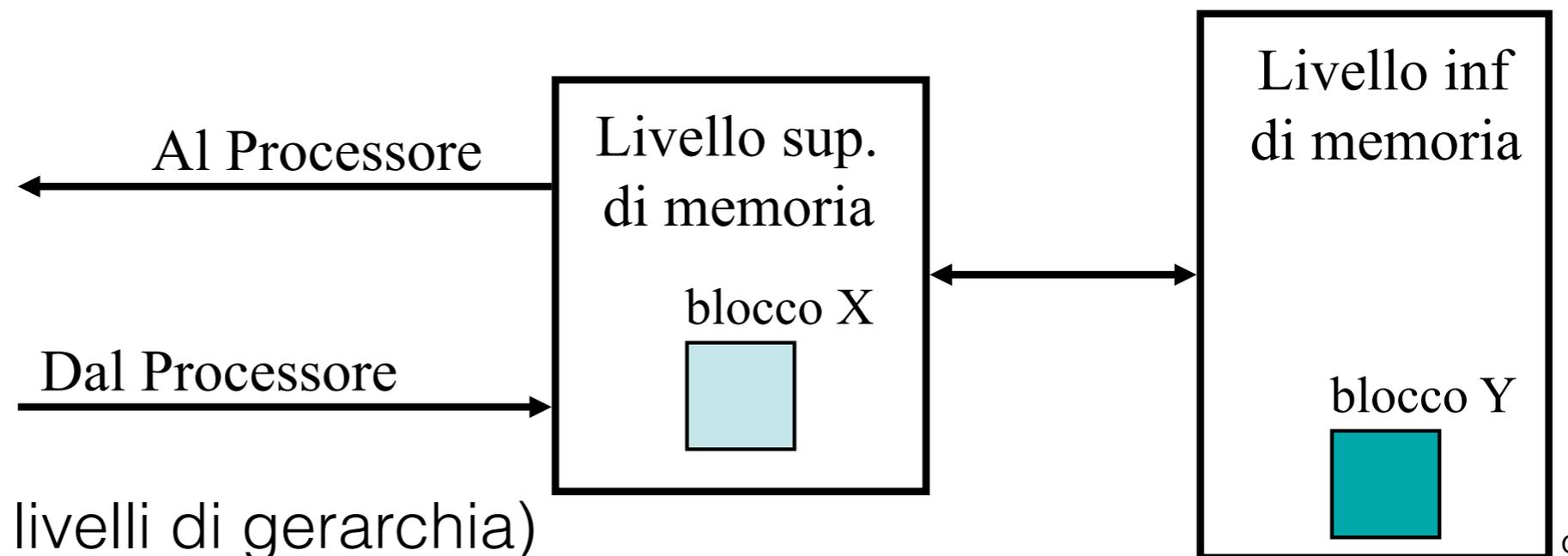
- Memoria al livello superiore della gerarchia, veloce ma piccola
- **Obiettivo:** fornire dati al processore in uno o due cicli di clock

Cache

- Memoria al livello superiore della gerarchia, veloce ma piccola
- **Obiettivo:** fornire dati al processore in uno o due cicli di clock
- **Idea:** Sfruttare il **principio di località** dei programmi e tenere in memoria cache i dati utilizzati più di recente

Cache

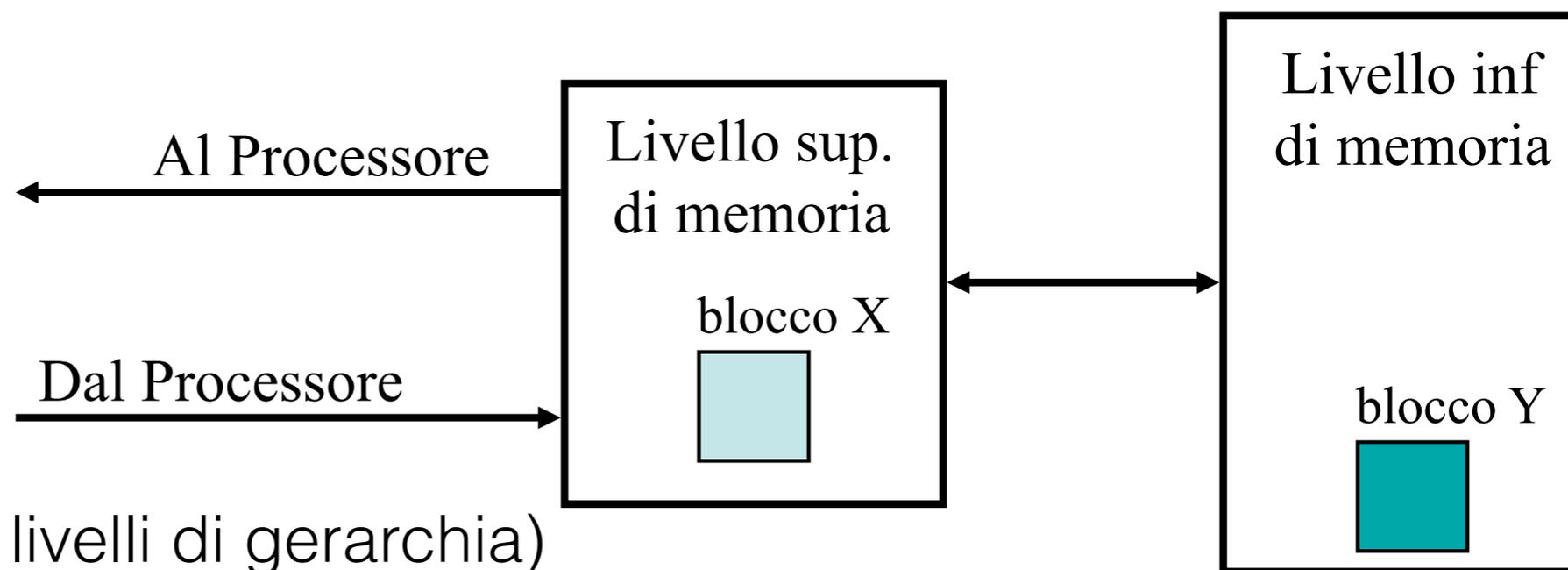
- Memoria al livello superiore della gerarchia, veloce ma piccola
- **Obiettivo:** fornire dati al processore in uno o due cicli di clock
- **Idea:** Sfruttare il **principio di località** dei programmi e tenere in memoria cache i dati utilizzati più di recente
- Il processore richiede un dato al sistema di memoria:



(Si considerino solo due livelli di gerarchia)

Cache

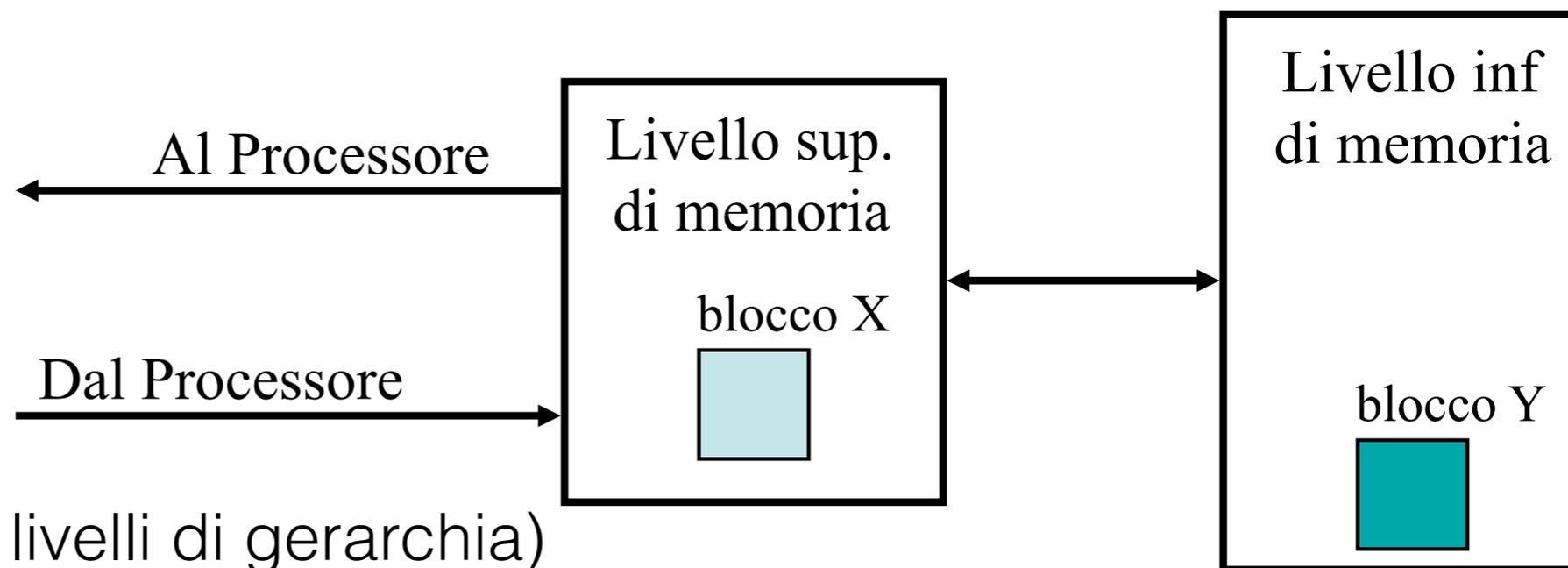
- Memoria al livello superiore della gerarchia, veloce ma piccola
- **Obiettivo:** fornire dati al processore in uno o due cicli di clock
- **Idea:** Sfruttare il **principio di località** dei programmi e tenere in memoria cache i dati utilizzati più di recente
- Il processore richiede un dato al sistema di memoria:
 - La richiesta viene *prima* inviata al livello di memoria **superiore** (più *vicino* al processore)



(Si considerino solo due livelli di gerarchia)

Cache

- Memoria al livello superiore della gerarchia, veloce ma piccola
- **Obiettivo**: fornire dati al processore in uno o due cicli di clock
- **Idea**: Sfruttare il **principio di località** dei programmi e tenere in memoria cache i dati utilizzati più di recente
- Il processore richiede un dato al sistema di memoria:
 - La richiesta viene *prima* inviata al livello di memoria **superiore** (più *vicino* al processore)
 - Se il dato non è presente nel livello **superiore** (*fallimento* della richiesta) la ricerca viene effettuata nel livello **inferiore**



(Si considerino solo due livelli di gerarchia)

Gerarchia di memoria: definizioni

Gerarchia di memoria: definizioni

- **Hit (successo)**: dati presenti in un blocco del livello superiore (esempio: Blocco X)

Gerarchia di memoria: definizioni

- **Hit (successo)**: dati presenti in un blocco del livello superiore (esempio: Blocco X)
- Hit **Rate** (“%” di successo): numero di accessi a memoria che trovano il dato nel livello superiore sul numero totale di accessi

Gerarchia di memoria: definizioni

- **Hit (successo)**: dati presenti in un blocco del livello superiore (esempio: Blocco X)
- Hit **Rate** (“%” di successo): numero di accessi a memoria che trovano il dato nel livello superiore sul numero totale di accessi
- Hit **Time (tempo** di successo): tempo per accedere al dato nel livello superiore della gerarchia

Gerarchia di memoria: definizioni

- **Hit (successo)**: dati presenti in un blocco del livello superiore (esempio: Blocco X)
 - Hit **Rate** (“%” di successo): numero di accessi a memoria che trovano il dato nel livello superiore sul numero totale di accessi
 - Hit **Time (tempo di successo)**: tempo per accedere al dato nel livello superiore della gerarchia
- **Miss (fallimento)**: i dati devono essere recuperati dal livello inferiore della memoria (Blocco Y)

Gerarchia di memoria: definizioni

- **Hit (successo)**: dati presenti in un blocco del livello superiore (esempio: Blocco X)
 - Hit **Rate** (“%” di successo): numero di accessi a memoria che trovano il dato nel livello superiore sul numero totale di accessi
 - Hit **Time (tempo di successo)**: tempo per accedere al dato nel livello superiore della gerarchia
- **Miss (fallimento)**: i dati devono essere recuperati dal livello inferiore della memoria (Blocco Y)
 - Miss **Rate** (“%” di fallimento) = $1 - (\text{Hit Rate})$

Gerarchia di memoria: definizioni

- **Hit (successo)**: dati presenti in un blocco del livello superiore (esempio: Blocco X)
 - Hit **Rate** (“%” di successo): numero di accessi a memoria che trovano il dato nel livello superiore sul numero totale di accessi
 - Hit **Time (tempo di successo)**: tempo per accedere al dato nel livello superiore della gerarchia
- **Miss (fallimento)**: i dati devono essere recuperati dal livello inferiore della memoria (Blocco Y)
 - Miss **Rate** (“%” di fallimento) = $1 - (\text{Hit Rate})$
 - Miss **Penalty (tempo di fallimento)**: tempo per determinare il MISS + tempo necessario a sostituire un blocco nel livello superiore + tempo per trasferire il blocco al processore

Gerarchia di memoria: definizioni

- **Hit (successo)**: dati presenti in un blocco del livello superiore (esempio: Blocco X)
 - Hit **Rate** (“%” di successo): numero di accessi a memoria che trovano il dato nel livello superiore sul numero totale di accessi
 - Hit **Time (tempo di successo)**: tempo per accedere al dato nel livello superiore della gerarchia
- **Miss (fallimento)**: i dati devono essere recuperati dal livello inferiore della memoria (Blocco Y)
 - Miss **Rate** (“%” di fallimento) = $1 - (\text{Hit Rate})$
 - Miss **Penalty (tempo di fallimento)**: tempo per determinare il MISS + tempo necessario a sostituire un blocco nel livello superiore + tempo per trasferire il blocco al processore

Tipicamente si ha: **Hit Time** \ll **Miss Penalty**

Cache e principio di località

Cache e principio di località

- **Tempo medio** di accesso in presenza di memoria cache:

Cache e principio di località

- **Tempo medio** di accesso in presenza di memoria cache:
 - semplicemente la **media pesata** con le probabilità

Cache e principio di località

- **Tempo medio** di accesso in presenza di memoria cache:
 - semplicemente la **media pesata** con le probabilità

$$\text{HitTime} * \text{HitRate} + \text{MissRate} * \text{MissPenalty}$$

Cache e principio di località

- **Tempo medio** di accesso in presenza di memoria cache:
 - semplicemente la **media pesata** con le probabilità

$$\text{HitTime} * \text{HitRate} + \text{MissRate} * \text{MissPenalty}$$

- Le memorie cache sfruttano il principio di **località spaziale** trasferendo dal livello inferiore della gerarchia più dati di quanti non ne siano stati strettamente richiesti (blocco o linea di cache)

Cache e principio di località

- **Tempo medio** di accesso in presenza di memoria cache:
 - semplicemente la **media pesata** con le probabilità

$$\text{HitTime} * \text{HitRate} + \text{MissRate} * \text{MissPenalty}$$

- Le memorie cache sfruttano il principio di **località spaziale** trasferendo dal livello inferiore della gerarchia più dati di quanti non ne siano stati strettamente richiesti (blocco o linea di cache)
- La **località temporale** viene sfruttata nella scelta del blocco da sostituire nella gestione di un fallimento (es: sostituire il blocco a cui si è fatto accesso meno di recente)

Cache e tempi di accesso (Es.3 - TE 01/09/2015)

Cache e tempi di accesso (Es.3 - TE 01/09/2015)

- Un calcolatore è dotato di un sistema di memoria centrale e memoria cache con le seguenti caratteristiche:

Cache e tempi di accesso (Es.3 - TE 01/09/2015)

- Un calcolatore è dotato di un sistema di memoria centrale e memoria cache con le seguenti caratteristiche:
 - **MissRate** = 25%

Cache e tempi di accesso (Es.3 - TE 01/09/2015)

- Un calcolatore è dotato di un sistema di memoria centrale e memoria cache con le seguenti caratteristiche:
 - **MissRate** = 25%
 - **MissPenalty** = 800 ns

Cache e tempi di accesso (Es.3 - TE 01/09/2015)

- Un calcolatore è dotato di un sistema di memoria centrale e memoria cache con le seguenti caratteristiche:
 - **MissRate** = 25%
 - **MissPenalty** = 800 ns
 - **HitTime** = 40ns

Cache e tempi di accesso (Es.3 - TE 01/09/2015)

- Un calcolatore è dotato di un sistema di memoria centrale e memoria cache con le seguenti caratteristiche:
 - **MissRate** = 25%
 - **MissPenalty** = 800 ns
 - **HitTime** = 40ns
- Rispondere alle seguenti domande (giustificando i risultati ottenuti con gli opportuni calcoli):

Cache e tempi di accesso (Es.3 - TE 01/09/2015)

- Un calcolatore è dotato di un sistema di memoria centrale e memoria cache con le seguenti caratteristiche:
 - **MissRate** = 25%
 - **MissPenalty** = 800 ns
 - **HitTime** = 40ns
- Rispondere alle seguenti domande (giustificando i risultati ottenuti con gli opportuni calcoli):
 - A. Calcolare il tempo medio di accesso ai dati del sistema con memoria cache.

Cache e tempi di accesso (Es.3 - TE 01/09/2015)

- Un calcolatore è dotato di un sistema di memoria centrale e memoria cache con le seguenti caratteristiche:
 - **MissRate** = 25%
 - **MissPenalty** = 800 ns
 - **HitTime** = 40ns
- Rispondere alle seguenti domande (giustificando i risultati ottenuti con gli opportuni calcoli):
 - A. Calcolare il tempo medio di accesso ai dati del sistema con memoria cache.
 - B. Mantenendo invariati Miss Rate e Hit Time, qual è il valore massimo che può avere Miss Penalty affinché il tempo medio sia inferiore a 200 ns?

Cache e tempi di accesso (Es.3 - TE 01/09/2015)

- Un calcolatore è dotato di un sistema di memoria centrale e memoria cache con le seguenti caratteristiche:
 - **MissRate** = 25%
 - **MissPenalty** = 800 ns
 - **HitTime** = 40ns
 - **HitRate** =
- Rispondere alle seguenti domande (giustificando i risultati ottenuti con gli opportuni calcoli):
 - A. Calcolare il tempo medio di accesso ai dati del sistema con memoria cache.
 - B. Mantenendo invariati Miss Rate e Hit Time, qual è il valore massimo che può avere Miss Penalty affinché il tempo medio sia inferiore a 200 ns?

Cache e tempi di accesso (Es.3 - TE 01/09/2015)

- Un calcolatore è dotato di un sistema di memoria centrale e memoria cache con le seguenti caratteristiche:
 - **MissRate** = 25%
 - **MissPenalty** = 800 ns
 - **HitTime** = 40ns
 - **HitRate** = 1 - **MissRate** =
- Rispondere alle seguenti domande (giustificando i risultati ottenuti con gli opportuni calcoli):
 - A. Calcolare il tempo medio di accesso ai dati del sistema con memoria cache.
 - B. Mantenendo invariati Miss Rate e Hit Time, qual è il valore massimo che può avere Miss Penalty affinché il tempo medio sia inferiore a 200 ns?

Cache e tempi di accesso (Es.3 - TE 01/09/2015)

- Un calcolatore è dotato di un sistema di memoria centrale e memoria cache con le seguenti caratteristiche:
 - **MissRate** = 25%
 - **MissPenalty** = 800 ns
 - **HitTime** = 40ns
 - **HitRate** = 1 - **MissRate** = 75%
- Rispondere alle seguenti domande (giustificando i risultati ottenuti con gli opportuni calcoli):
 - A. Calcolare il tempo medio di accesso ai dati del sistema con memoria cache.
 - B. Mantenendo invariati Miss Rate e Hit Time, qual è il valore massimo che può avere Miss Penalty affinché il tempo medio sia inferiore a 200 ns?

Cache e tempi di accesso (Es.3 - TE 01/09/2015)

- Un calcolatore è dotato di un sistema di memoria centrale e memoria cache con le seguenti caratteristiche:
 - **MissRate** = 25%
 - **MissPenalty** = 800 ns
 - **HitTime** = 40ns
 - **HitRate** = 1 - **MissRate** = 75%
- Rispondere alle seguenti domande (giustificando i risultati ottenuti con gli opportuni calcoli):
 - A. Calcolare il tempo medio di accesso ai dati del sistema con memoria cache.

Tempo medio =
 - B. Mantenendo invariati Miss Rate e Hit Time, qual è il valore massimo che può avere Miss Penalty affinché il tempo medio sia inferiore a 200 ns?

Cache e tempi di accesso (Es.3 - TE 01/09/2015)

- Un calcolatore è dotato di un sistema di memoria centrale e memoria cache con le seguenti caratteristiche:
 - **MissRate** = 25%
 - **MissPenalty** = 800 ns
 - **HitTime** = 40ns
 - **HitRate** = 1 - **MissRate** = 75%
- Rispondere alle seguenti domande (giustificando i risultati ottenuti con gli opportuni calcoli):
 - A. Calcolare il tempo medio di accesso ai dati del sistema con memoria cache.

Tempo medio = **HitTime** +
 - B. Mantenendo invariati Miss Rate e Hit Time, qual è il valore massimo che può avere Miss Penalty affinché il tempo medio sia inferiore a 200 ns?

Cache e tempi di accesso (Es.3 - TE 01/09/2015)

- Un calcolatore è dotato di un sistema di memoria centrale e memoria cache con le seguenti caratteristiche:
 - **MissRate** = 25%
 - **MissPenalty** = 800 ns
 - **HitTime** = 40ns
 - **HitRate** = 1 - **MissRate** = 75%
- Rispondere alle seguenti domande (giustificando i risultati ottenuti con gli opportuni calcoli):
 - A. Calcolare il tempo medio di accesso ai dati del sistema con memoria cache.

Tempo medio = **HitTime** + **MissPenalty**

- B. Mantenendo invariati Miss Rate e Hit Time, qual è il valore massimo che può avere Miss Penalty affinché il tempo medio sia inferiore a 200 ns?

Cache e tempi di accesso (Es.3 - TE 01/09/2015)

- Un calcolatore è dotato di un sistema di memoria centrale e memoria cache con le seguenti caratteristiche:
 - **MissRate** = 25%
 - **MissPenalty** = 800 ns
 - **HitTime** = 40ns
 - **HitRate** = 1 - **MissRate** = 75%
- Rispondere alle seguenti domande (giustificando i risultati ottenuti con gli opportuni calcoli):
 - A. Calcolare il tempo medio di accesso ai dati del sistema con memoria cache.

$$\text{Tempo medio} = \text{HitRate} * \text{HitTime} + \text{MissPenalty}$$

- B. Mantenendo invariati Miss Rate e Hit Time, qual è il valore massimo che può avere Miss Penalty affinché il tempo medio sia inferiore a 200 ns?

Cache e tempi di accesso (Es.3 - TE 01/09/2015)

- Un calcolatore è dotato di un sistema di memoria centrale e memoria cache con le seguenti caratteristiche:
 - **MissRate** = 25%
 - **MissPenalty** = 800 ns
 - **HitTime** = 40ns
 - **HitRate** = 1 - **MissRate** = 75%
- Rispondere alle seguenti domande (giustificando i risultati ottenuti con gli opportuni calcoli):
 - A. Calcolare il tempo medio di accesso ai dati del sistema con memoria cache.
$$\text{Tempo medio} = \text{HitRate} * \text{HitTime} + \text{MissRate} * \text{MissPenalty}$$
 - B. Mantenendo invariati Miss Rate e Hit Time, qual è il valore massimo che può avere Miss Penalty affinché il tempo medio sia inferiore a 200 ns?

Cache e tempi di accesso (Es.3 - TE 01/09/2015)

- Un calcolatore è dotato di un sistema di memoria centrale e memoria cache con le seguenti caratteristiche:
 - **MissRate** = 25%
 - **MissPenalty** = 800 ns
 - **HitTime** = 40ns
 - **HitRate** = 1 - **MissRate** = 75%
- Rispondere alle seguenti domande (giustificando i risultati ottenuti con gli opportuni calcoli):
 - A. Calcolare il tempo medio di accesso ai dati del sistema con memoria cache.

$$\text{Tempo medio} = 0.75 * 40\text{ns} + 0.25 * 800\text{ns} = 30 + 200 = \mathbf{230 \text{ ns}}$$

- B. Mantenendo invariati Miss Rate e Hit Time, qual è il valore massimo che può avere Miss Penalty affinché il tempo medio sia inferiore a 200 ns?

Cache e tempi di accesso (Es.3 - TE 01/09/2015)

- Un calcolatore è dotato di un sistema di memoria centrale e memoria cache con le seguenti caratteristiche:
 - **MissRate** = 25%
 - **MissPenalty** = 800 ns
 - **HitTime** = 40ns
 - **HitRate** = 1 - **MissRate** = 75%
- Rispondere alle seguenti domande (giustificando i risultati ottenuti con gli opportuni calcoli):
 - A. Calcolare il tempo medio di accesso ai dati del sistema con memoria cache.

$$\text{Tempo medio} = 0.75 * 40\text{ns} + 0.25 * 800\text{ns} = 30 + 200 = \mathbf{230 \text{ ns}}$$

- B. Mantenendo invariati Miss Rate e Hit Time, qual è il valore massimo che può avere Miss Penalty affinché il tempo medio sia inferiore a 200 ns?

$$0.75 * 40 \text{ ns} + 0.25 * x < 200 \text{ ns}$$

Cache e tempi di accesso (Es.3 - TE 01/09/2015)

- Un calcolatore è dotato di un sistema di memoria centrale e memoria cache con le seguenti caratteristiche:
 - **MissRate** = 25%
 - **MissPenalty** = 800 ns
 - **HitTime** = 40ns
 - **HitRate** = 1 - **MissRate** = 75%
- Rispondere alle seguenti domande (giustificando i risultati ottenuti con gli opportuni calcoli):
 - A. Calcolare il tempo medio di accesso ai dati del sistema con memoria cache.

$$\text{Tempo medio} = \mathbf{0.75 * 40ns} + \mathbf{0.25 * 800ns} = 30 + 200 = \mathbf{230 ns}$$

- B. Mantenendo invariati Miss Rate e Hit Time, qual è il valore massimo che può avere Miss Penalty affinché il tempo medio sia inferiore a 200 ns?

$$\mathbf{0.75 * 40 ns} + \mathbf{0.25 * x} < 200 \text{ ns}$$

$$\mathbf{30 ns} + \mathbf{0.25 * x} < 200 \text{ ns}$$

Cache e tempi di accesso (Es.3 - TE 01/09/2015)

- Un calcolatore è dotato di un sistema di memoria centrale e memoria cache con le seguenti caratteristiche:
 - **MissRate** = 25%
 - **MissPenalty** = 800 ns
 - **HitTime** = 40ns
 - **HitRate** = 1 - **MissRate** = 75%
- Rispondere alle seguenti domande (giustificando i risultati ottenuti con gli opportuni calcoli):

A. Calcolare il tempo medio di accesso ai dati del sistema con memoria cache.

$$\text{Tempo medio} = \mathbf{0.75*40ns} + \mathbf{0.25*800ns} = 30 + 200 = \mathbf{230 ns}$$

B. Mantenendo invariati Miss Rate e Hit Time, qual è il valore massimo che può avere Miss Penalty affinché il tempo medio sia inferiore a 200 ns?

$$\begin{aligned} \mathbf{0.75*40 ns} + \mathbf{0.25*x} &< 200 \text{ ns} \\ \mathbf{30 ns} + \mathbf{0.25*x} &< 200 \text{ ns} \\ \mathbf{x} &< 170/0.25 \text{ ns} \end{aligned}$$

Cache e tempi di accesso (Es.3 - TE 01/09/2015)

- Un calcolatore è dotato di un sistema di memoria centrale e memoria cache con le seguenti caratteristiche:
 - **MissRate** = 25%
 - **MissPenalty** = 800 ns
 - **HitTime** = 40ns
 - **HitRate** = 1 - **MissRate** = 75%
- Rispondere alle seguenti domande (giustificando i risultati ottenuti con gli opportuni calcoli):
 - A. Calcolare il tempo medio di accesso ai dati del sistema con memoria cache.

$$\text{Tempo medio} = \mathbf{0.75*40ns} + \mathbf{0.25*800ns} = 30 + 200 = \mathbf{230 ns}$$

- B. Mantenendo invariati Miss Rate e Hit Time, qual è il valore massimo che può avere Miss Penalty affinché il tempo medio sia inferiore a 200 ns?

$$\begin{aligned}\mathbf{0.75*40 ns} + \mathbf{0.25*x} &< 200 \text{ ns} \\ \mathbf{30 ns} + \mathbf{0.25*x} &< 200 \text{ ns} \\ \mathbf{x} &< 170/0.25 \text{ ns} \\ \mathbf{x} &< \mathbf{680 ns}\end{aligned}$$

Memoria fisica e memoria virtuale (Es.3 - TE ?)

Parte A

- Un sistema dispone di 64 Kbyte di memoria fisica indirizzabile e di 128 Kbyte di memoria virtuale indirizzabile; inoltre la memoria virtuale è organizzata in pagine di 512 byte.
- Rispondere alle seguenti domande (giustificando i risultati ottenuti con gli opportuni calcoli):
 1. Quante sono le pagine di memoria virtuale?
 2. Definire la struttura dell'indirizzo virtuale e di quello fisico indicando la lunghezza dei campi che li costituiscono.

Memoria fisica e memoria virtuale (Es.3 - TE ?)

Parte A

- Un sistema dispone di 64 Kbyte di memoria fisica indirizzabile e di 128 Kbyte di memoria virtuale indirizzabile; inoltre la memoria virtuale è organizzata in pagine di 512 byte.
- Rispondere alle seguenti domande (giustificando i risultati ottenuti con gli opportuni calcoli):
 1. Quante sono le pagine di memoria virtuale?

Memoria fisica e memoria virtuale (Es.3 - TE ?)

Parte A

- Un sistema dispone di 64 Kbyte di memoria fisica indirizzabile e di 128 Kbyte di memoria virtuale indirizzabile; inoltre la memoria virtuale è organizzata in pagine di 512 byte.
- Rispondere alle seguenti domande (giustificando i risultati ottenuti con gli opportuni calcoli):
 1. Quante sono le pagine di memoria virtuale?
64 Kbyte memoria **fisica** = $64 * 1024$ byte

Memoria fisica e memoria virtuale (Es.3 - TE ?)

Parte A

- Un sistema dispone di 64 Kbyte di memoria fisica indirizzabile e di 128 Kbyte di memoria virtuale indirizzabile; inoltre la memoria virtuale è organizzata in pagine di 512 byte.
- Rispondere alle seguenti domande (giustificando i risultati ottenuti con gli opportuni calcoli):
 1. Quante sono le pagine di memoria virtuale?
 - 64 Kbyte memoria **fisica** = $64 * 1024$ byte
 - 128 Kbyte memoria **virtuale**

Memoria fisica e memoria virtuale (Es.3 - TE ?)

Parte A

- Un sistema dispone di 64 Kbyte di memoria fisica indirizzabile e di 128 Kbyte di memoria virtuale indirizzabile; inoltre la memoria virtuale è organizzata in pagine di 512 byte.
- Rispondere alle seguenti domande (giustificando i risultati ottenuti con gli opportuni calcoli):
 1. Quante sono le pagine di memoria virtuale?
 - 64 Kbyte memoria **fisica** = $64 * 1024$ byte
 - 128 Kbyte memoria **virtuale**
 - 512 byte per **pagina** = 0,5 Kbyte

Memoria fisica e memoria virtuale (Es.3 - TE ?)

Parte A

- Un sistema dispone di 64 Kbyte di memoria fisica indirizzabile e di 128 Kbyte di memoria virtuale indirizzabile; inoltre la memoria virtuale è organizzata in pagine di 512 byte.

- Rispondere alle seguenti domande (giustificando i risultati ottenuti con gli opportuni calcoli):

1. Quante sono le pagine di memoria virtuale?

64 Kbyte memoria **fisica** = $64 * 1024$ byte

128 Kbyte memoria **virtuale**

512 byte per **pagina** = 0,5 Kbyte

$64 / 0,5 = 128$ pagine di memoria **fisica**

Memoria fisica e memoria virtuale (Es.3 - TE ?)

Parte A

- Un sistema dispone di 64 Kbyte di memoria fisica indirizzabile e di 128 Kbyte di memoria virtuale indirizzabile; inoltre la memoria virtuale è organizzata in pagine di 512 byte.

- Rispondere alle seguenti domande (giustificando i risultati ottenuti con gli opportuni calcoli):

1. Quante sono le pagine di memoria virtuale?

64 Kbyte memoria **fisica** = $64 * 1024$ byte

128 Kbyte memoria **virtuale**

512 byte per **pagina** = 0,5 Kbyte

$64 / 0,5 = 128$ pagine di memoria **fisica**

$128 / 0,5 = 256$ pagine di memoria **virtuale**

Memoria fisica e memoria virtuale (Es.3 - TE ?)

2. Definire la struttura dell'indirizzo virtuale e di quello fisico indicando la lunghezza dei campi che li costituiscono.

Memoria fisica e memoria virtuale (Es.3 - TE ?)

2. Definire la struttura dell'indirizzo virtuale e di quello fisico indicando la lunghezza dei campi che li costituiscono.

512 byte per **pagina** = 0,5 Kbyte

Memoria fisica e memoria virtuale (Es.3 - TE ?)

2. Definire la struttura dell'indirizzo virtuale e di quello fisico indicando la lunghezza dei campi che li costituiscono.

512 byte per **pagina** = 0,5 Kbyte

- ogni byte deve essere singolarmente indirizzabile

Memoria fisica e memoria virtuale (Es.3 - TE ?)

2. Definire la struttura dell'indirizzo virtuale e di quello fisico indicando la lunghezza dei campi che li costituiscono.

512 byte per **pagina** = 0,5 Kbyte

- ogni byte deve essere singolarmente indirizzabile
- 512 “valori” diversi con una sola parola binaria

Memoria fisica e memoria virtuale (Es.3 - TE ?)

2. Definire la struttura dell'indirizzo virtuale e di quello fisico indicando la lunghezza dei campi che li costituiscono.

512 byte per **pagina** = 0,5 Kbyte

- ogni byte deve essere singolarmente indirizzabile
- 512 “valori” diversi con una sola parola binaria
 - ho bisogno di $\text{ceil}(\log(2, 512)) = \mathbf{9 \text{ bit}}$

Memoria fisica e memoria virtuale (Es.3 - TE ?)

2. Definire la struttura dell'indirizzo virtuale e di quello fisico indicando la lunghezza dei campi che li costituiscono.

512 byte per **pagina** = 0,5 Kbyte

- ogni byte deve essere singolarmente indirizzabile
- 512 “valori” diversi con una sola parola binaria
 - ho bisogno di $\text{ceil}(\log(2, 512)) = \mathbf{9 \text{ bit}}$

$64/0,5 = 128$ pagine di memoria **fisica**

Memoria fisica e memoria virtuale (Es.3 - TE ?)

2. Definire la struttura dell'indirizzo virtuale e di quello fisico indicando la lunghezza dei campi che li costituiscono.

512 byte per **pagina** = 0,5 Kbyte

- ogni byte deve essere singolarmente indirizzabile
- 512 “valori” diversi con una sola parola binaria
 - ho bisogno di $\text{ceil}(\log(2, 512)) = \mathbf{9 \text{ bit}}$

$64/0,5 = 128$ pagine di memoria **fisica**

- devo identificare ognuna delle 128 pagine di memoria virtuale

Memoria fisica e memoria virtuale (Es.3 - TE ?)

2. Definire la struttura dell'indirizzo virtuale e di quello fisico indicando la lunghezza dei campi che li costituiscono.

512 byte per **pagina** = 0,5 Kbyte

- ogni byte deve essere singolarmente indirizzabile
- 512 “valori” diversi con una sola parola binaria
 - ho bisogno di $\text{ceil}(\log(2, 512)) = \mathbf{9 \text{ bit}}$

$64/0,5 = 128$ pagine di memoria **fisica**

- devo identificare ognuna delle 128 pagine di memoria virtuale
 - ho bisogno di $\text{ceil}(\log(2, 128)) = \mathbf{7 \text{ bit}}$

Memoria fisica e memoria virtuale (Es.3 - TE ?)

2. Definire la struttura dell'indirizzo virtuale e di quello fisico indicando la lunghezza dei campi che li costituiscono.

512 byte per **pagina** = 0,5 Kbyte

- ogni byte deve essere singolarmente indirizzabile
- 512 “valori” diversi con una sola parola binaria
 - ho bisogno di $\text{ceil}(\log(2, 512)) = \mathbf{9 \text{ bit}}$

$64/0,5 = 128$ pagine di memoria **fisica**

- devo identificare ognuna delle 128 pagine di memoria virtuale
 - ho bisogno di $\text{ceil}(\log(2, 128)) = \mathbf{7 \text{ bit}}$

$128/0,5 = 256$ pagine di memoria **virtuale**

Memoria fisica e memoria virtuale (Es.3 - TE ?)

2. Definire la struttura dell'indirizzo virtuale e di quello fisico indicando la lunghezza dei campi che li costituiscono.

512 byte per **pagina** = 0,5 Kbyte

- ogni byte deve essere singolarmente indirizzabile
- 512 “valori” diversi con una sola parola binaria
 - ho bisogno di $\text{ceil}(\log(2, 512)) = \mathbf{9 \text{ bit}}$

$64/0,5 = 128$ pagine di memoria **fisica**

- devo identificare ognuna delle 128 pagine di memoria virtuale
 - ho bisogno di $\text{ceil}(\log(2, 128)) = \mathbf{7 \text{ bit}}$

$128/0,5 = 256$ pagine di memoria **virtuale**

- devo identificare ognuna delle 256 pagine di memoria virtuale

Memoria fisica e memoria virtuale (Es.3 - TE ?)

2. Definire la struttura dell'indirizzo virtuale e di quello fisico indicando la lunghezza dei campi che li costituiscono.

512 byte per **pagina** = 0,5 Kbyte

- ogni byte deve essere singolarmente indirizzabile
- 512 “valori” diversi con una sola parola binaria
 - ho bisogno di $\text{ceil}(\log(2, 512)) = \mathbf{9 \text{ bit}}$

$64/0,5 = 128$ pagine di memoria **fisica**

- devo identificare ognuna delle 128 pagine di memoria virtuale
 - ho bisogno di $\text{ceil}(\log(2, 128)) = \mathbf{7 \text{ bit}}$

$128/0,5 = 256$ pagine di memoria **virtuale**

- devo identificare ognuna delle 256 pagine di memoria virtuale
 - ho bisogno di $\text{ceil}(\log(2, 256)) = \mathbf{8 \text{ bit}}$

Memoria fisica e memoria virtuale (Es.3 - TE ?)

2. Definire la struttura dell'indirizzo virtuale e di quello fisico indicando la lunghezza dei campi che li costituiscono.

512 byte per **pagina** = 0,5 Kbyte

- ogni byte deve essere singolarmente indirizzabile
- 512 “valori” diversi con una sola parola binaria
 - ho bisogno di $\text{ceil}(\log(2, 512)) = \mathbf{9 \text{ bit}}$

64/0,5 = 128 pagine di memoria **fisica**

- devo identificare ognuna delle 128 pagine di memoria virtuale
 - ho bisogno di $\text{ceil}(\log(2, 128)) = \mathbf{7 \text{ bit}}$

Indirizzo fisico = 9+7 = **16 bit**

128/0,5 = 256 pagine di memoria **virtuale**

- devo identificare ognuna delle 256 pagine di memoria virtuale
 - ho bisogno di $\text{ceil}(\log(2, 256)) = \mathbf{8 \text{ bit}}$

Memoria fisica e memoria virtuale (Es.3 - TE ?)

2. Definire la struttura dell'indirizzo virtuale e di quello fisico indicando la lunghezza dei campi che li costituiscono.

512 byte per **pagina** = 0,5 Kbyte

- ogni byte deve essere singolarmente indirizzabile
- 512 “valori” diversi con una sola parola binaria
 - ho bisogno di $\text{ceil}(\log(2, 512)) = \mathbf{9 \text{ bit}}$

64/0,5 = 128 pagine di memoria **fisica**

- devo identificare ognuna delle 128 pagine di memoria virtuale
 - ho bisogno di $\text{ceil}(\log(2, 128)) = \mathbf{7 \text{ bit}}$

Indirizzo fisico = 9+7 = **16 bit**

128/0,5 = 256 pagine di memoria **virtuale**

- devo identificare ognuna delle 256 pagine di memoria virtuale
 - ho bisogno di $\text{ceil}(\log(2, 256)) = \mathbf{8 \text{ bit}}$

Indirizzo virtuale = 9+8 = **17 bit**

Memoria fisica e memoria virtuale (Es.3 - TE ?)

Parte B

- Si consideri un sistema con le seguenti caratteristiche:
 - Indirizzo virtuale di 8 bit
 - Indirizzo fisico di 7 bit
 - Dimensione pagine 16 byte

- L'indirizzo virtuale 00101000 può corrispondere all'indirizzo fisico 0010100? Giustificare la risposta.

Memoria fisica e memoria virtuale (Es.3 - TE ?)

Parte B

- Si consideri un sistema con le seguenti caratteristiche:
 - Indirizzo virtuale di 8 bit
 - Indirizzo fisico di 7 bit
 - Dimensione pagine 16 byte
 - ogni byte deve essere singolarmente indirizzabile

- L'indirizzo virtuale 00101000 può corrispondere all'indirizzo fisico 0010100? Giustificare la risposta.

Memoria fisica e memoria virtuale (Es.3 - TE ?)

Parte B

- Si consideri un sistema con le seguenti caratteristiche:
 - Indirizzo virtuale di 8 bit
 - Indirizzo fisico di 7 bit
 - Dimensione pagine 16 byte
 - ogni byte deve essere singolarmente indirizzabile
 - 16 “valori” diversi con una sola parola binaria
- L'indirizzo virtuale 00101000 può corrispondere all'indirizzo fisico 0010100? Giustificare la risposta.

Memoria fisica e memoria virtuale (Es.3 - TE ?)

Parte B

- Si consideri un sistema con le seguenti caratteristiche:
 - Indirizzo virtuale di 8 bit
 - Indirizzo fisico di 7 bit
 - Dimensione pagine 16 byte
 - ogni byte deve essere singolarmente indirizzabile
 - 16 “valori” diversi con una sola parola binaria
 - ho bisogno di $\text{ceil}(\log(2, 16)) = \mathbf{4 \text{ bit}}$
- L'indirizzo virtuale 00101000 può corrispondere all'indirizzo fisico 0010100? Giustificare la risposta.

Memoria fisica e memoria virtuale (Es.3 - TE ?)

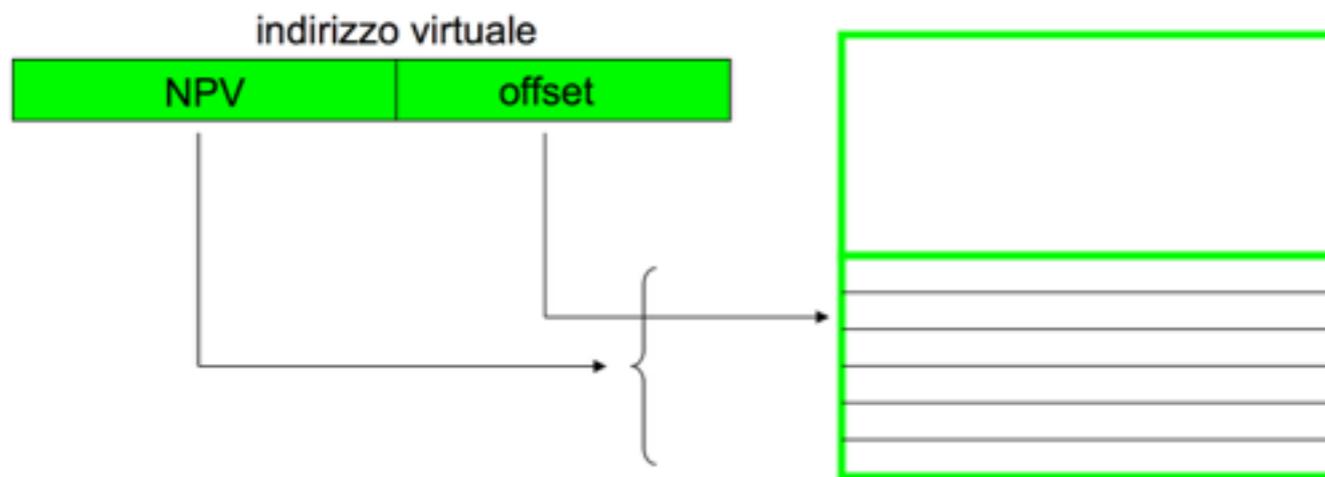
Parte B

- Si consideri un sistema con le seguenti caratteristiche:
 - Indirizzo virtuale di 8 bit
 - Indirizzo fisico di 7 bit
 - Dimensione pagine 16 byte
 - ogni byte deve essere singolarmente indirizzabile
 - 16 “valori” diversi con una sola parola binaria
 - ho bisogno di $\text{ceil}(\log(2, 16)) = \mathbf{4 \text{ bit}}$
- L'indirizzo virtuale **00101000** può corrispondere all'indirizzo fisico **0010100**? Giustificare la risposta.

Memoria fisica e memoria virtuale (Es.3 - TE ?)

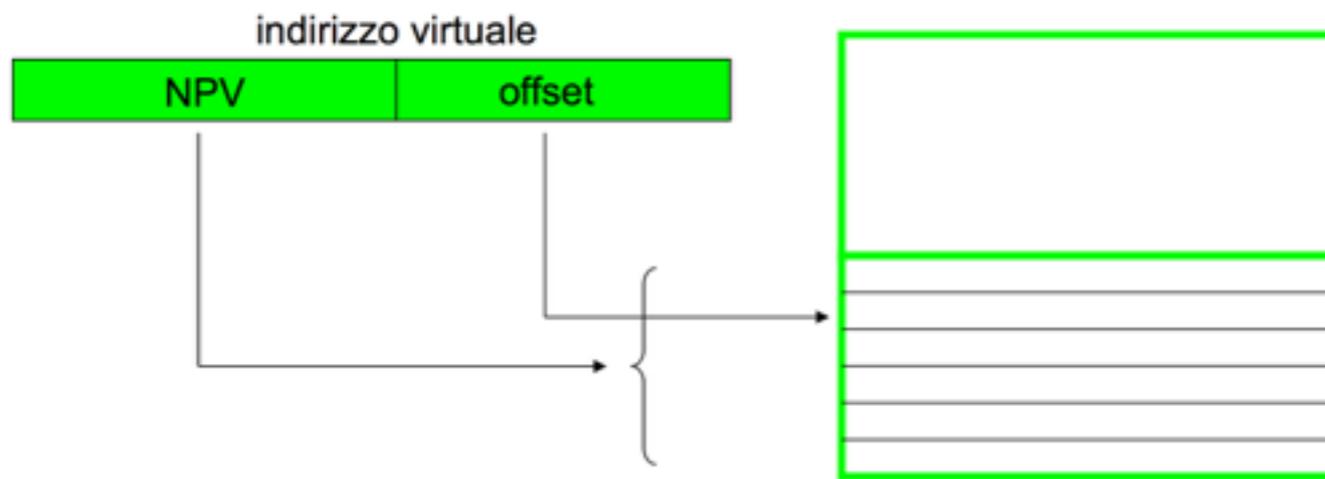
Memoria fisica e memoria virtuale (Es.3 - TE ?)

- Un indirizzo virtuale è costituito da un numero di pagina virtuale (NPV) e da uno spiazzamento (offset) all'interno della pagina

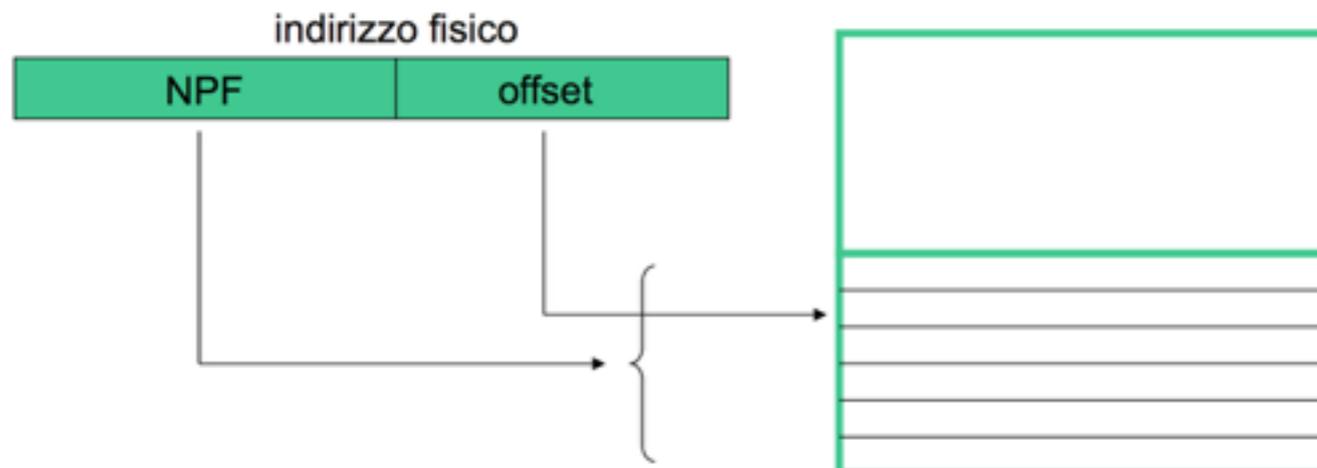


Memoria fisica e memoria virtuale (Es.3 - TE ?)

- Un indirizzo virtuale è costituito da un numero di pagina virtuale (NPV) e da uno spiazzamento (offset) all'interno della pagina

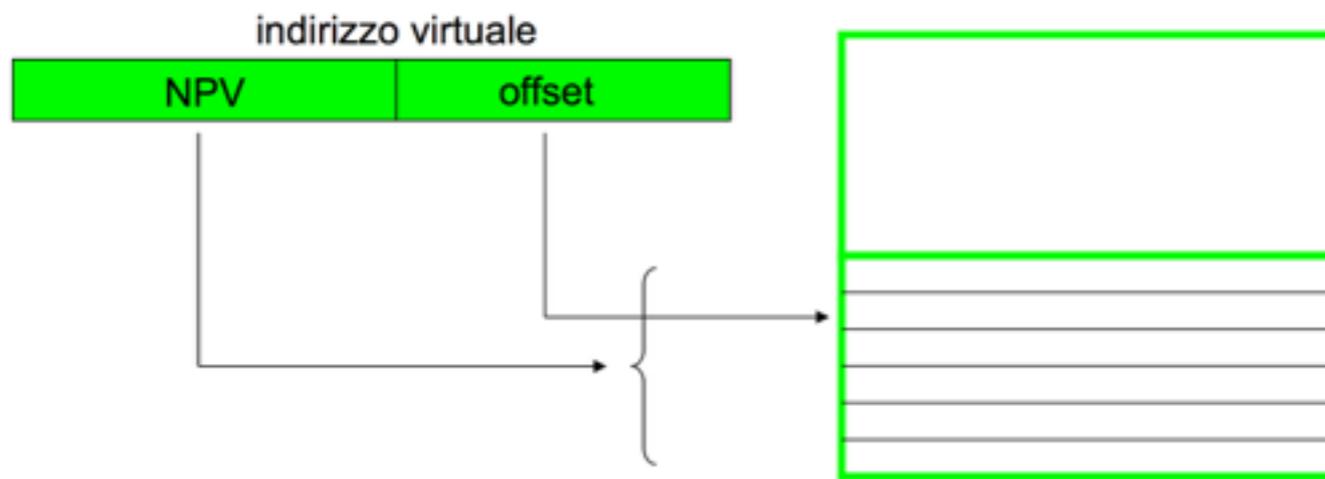


- E' del tutto analoga: si hanno un numero di pagina fisica (NPF) e da uno spiazzamento (offset) all'interno della pagina

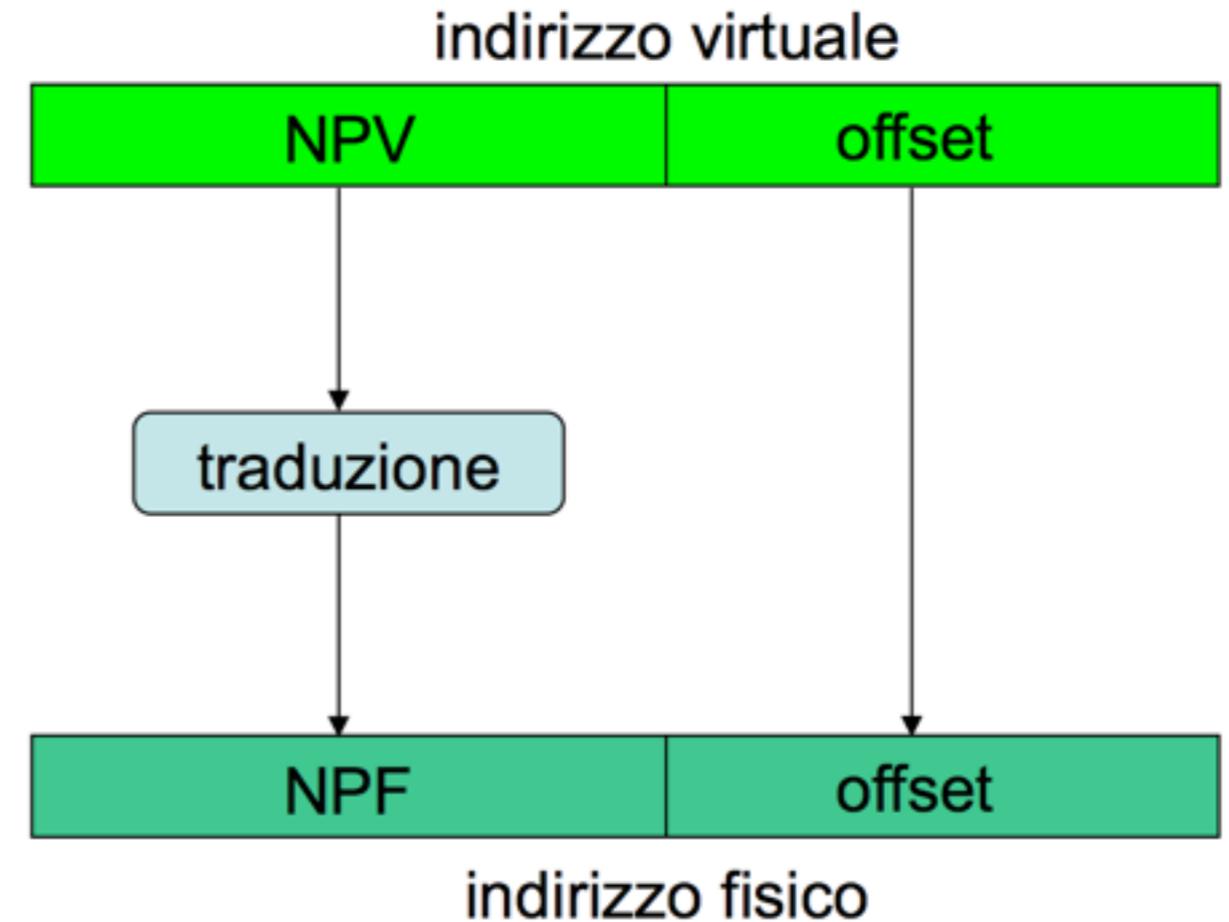
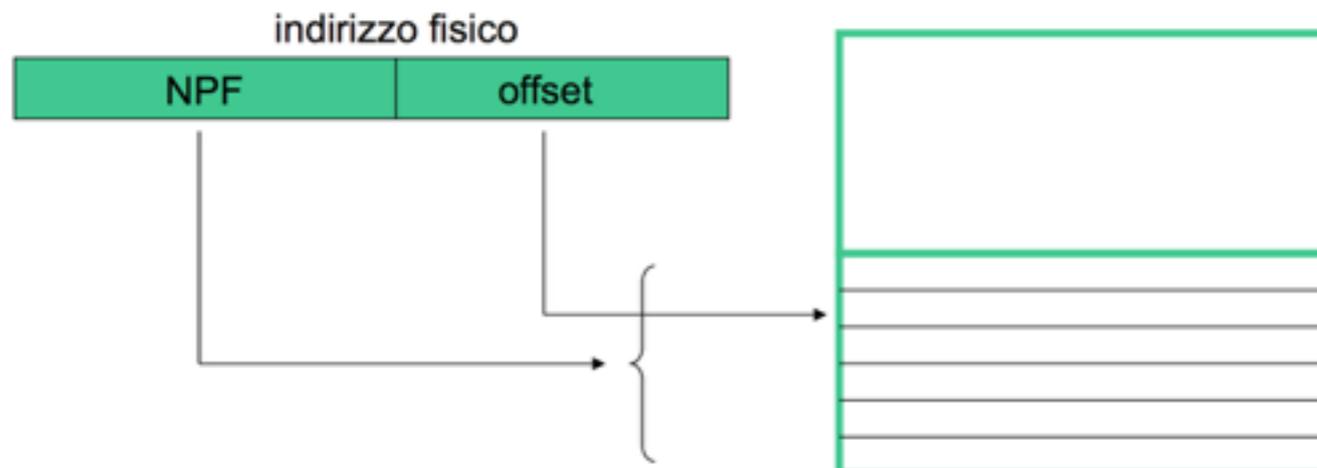


Memoria fisica e memoria virtuale (Es.3 - TE ?)

- Un indirizzo virtuale è costituito da un numero di pagina virtuale (NPV) e da uno spiazzamento (offset) all'interno della pagina

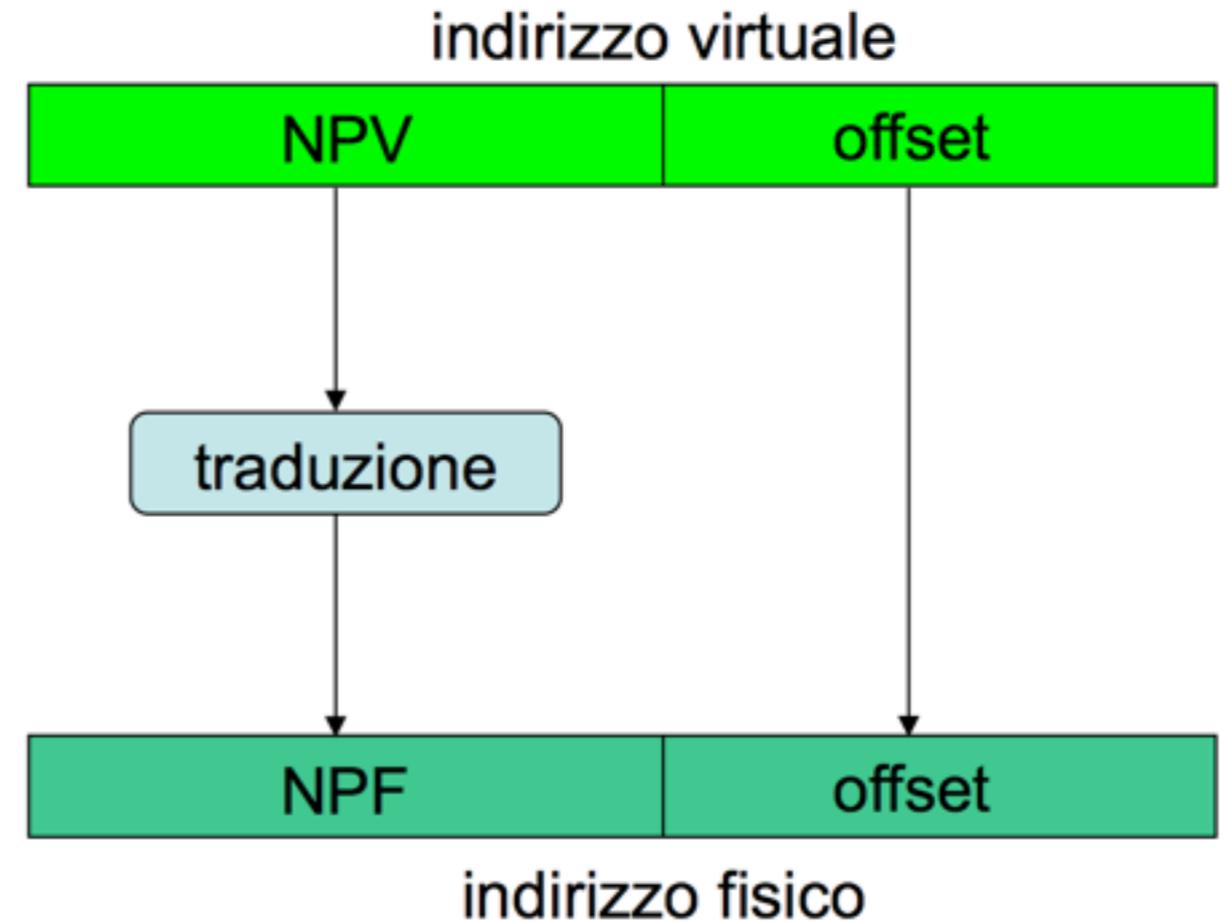


- E' del tutto analoga: si hanno un numero di pagina fisica (NPF) e da uno spiazzamento (offset) all'interno della pagina



le pagine virtuali e quelle fisiche hanno la stessa dimensione, quindi l'offset è lo stesso

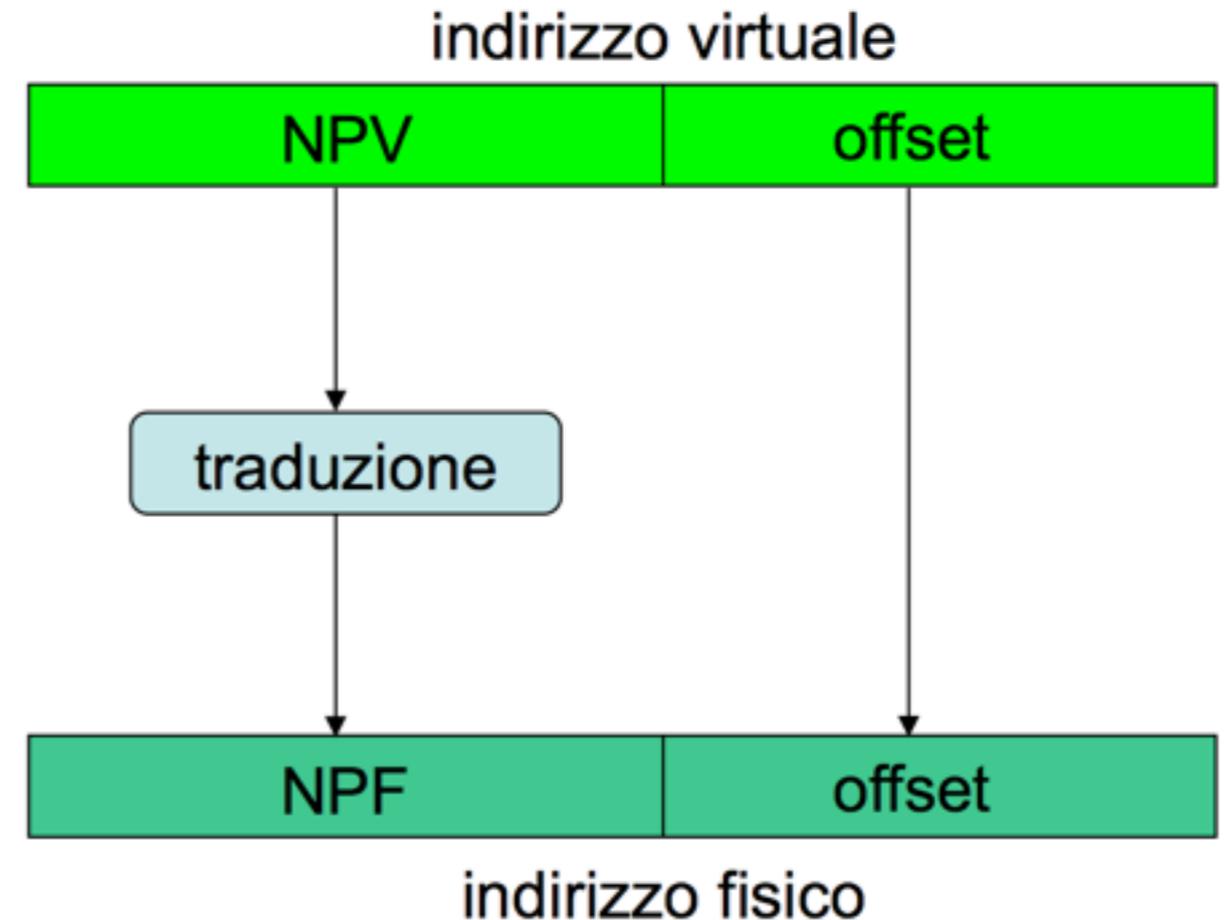
Memoria fisica e memoria virtuale (Es.3 - TE ?)



le pagine virtuali e quelle fisiche hanno la stessa dimensione, quindi l'offset è lo stesso

Memoria fisica e memoria virtuale (Es.3 - TE ?)

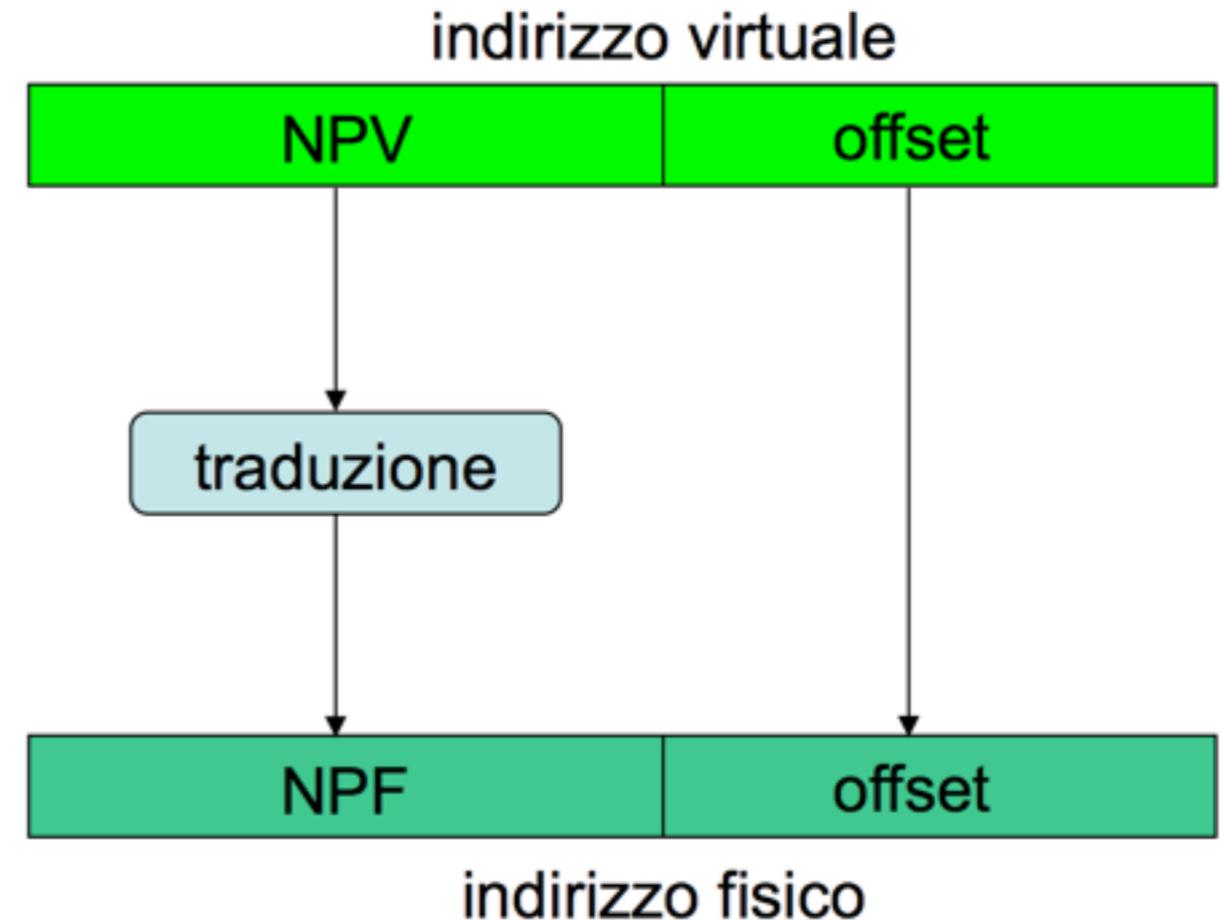
- Dimensione pagine 16 byte
-> **4 bit**



le pagine virtuali e quelle fisiche hanno la stessa dimensione, quindi l'offset è lo stesso

Memoria fisica e memoria virtuale (Es.3 - TE ?)

- Dimensione pagine 16 byte
-> **4 bit**
- L'indirizzo virtuale **00101000**
può corrispondere all'indirizzo
fisico **0010100**?

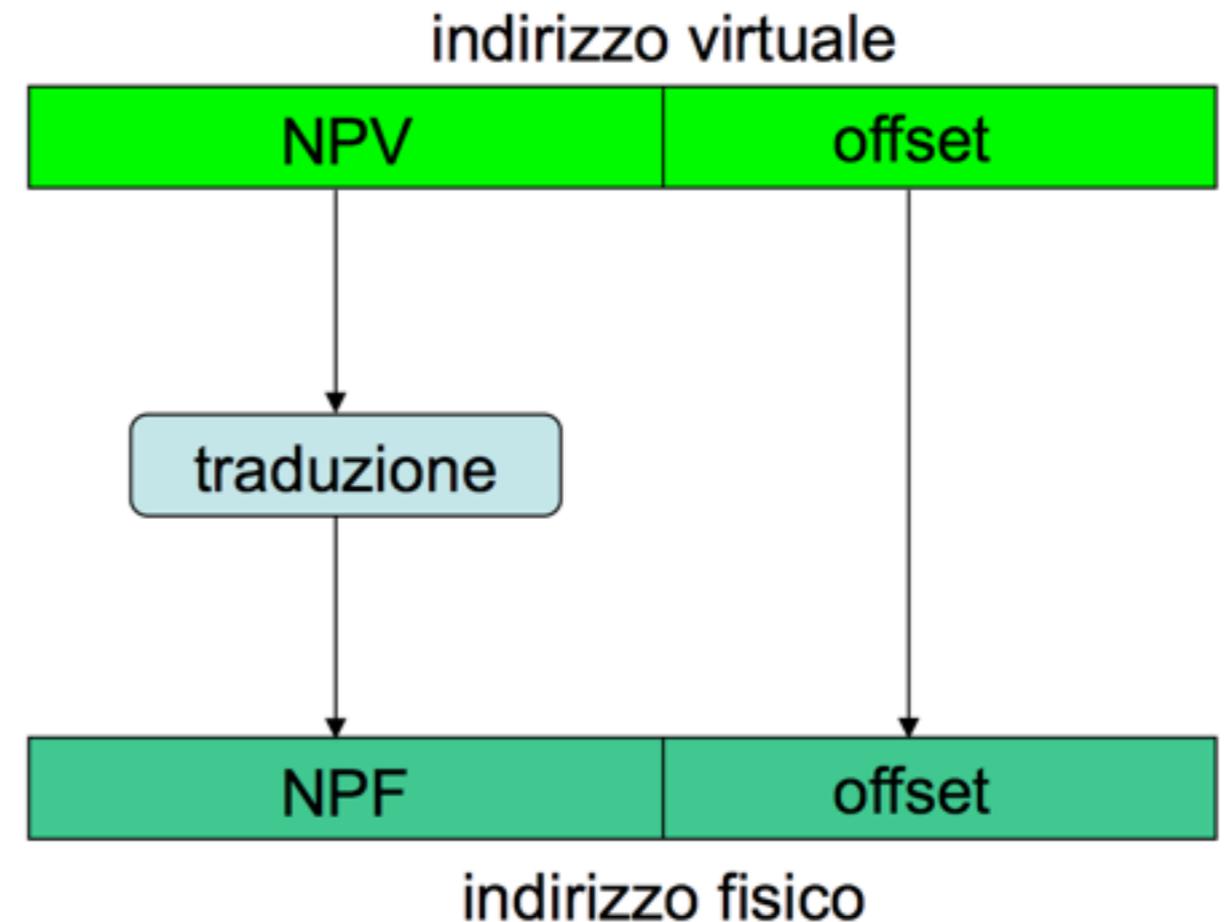


le pagine virtuali e quelle fisiche hanno la stessa dimensione, quindi l'offset è lo stesso

Memoria fisica e memoria virtuale (Es.3 - TE ?)

- Dimensione pagine 16 byte
-> **4 bit**
- L'indirizzo virtuale **00101000**
può corrispondere all'indirizzo
fisico **0010100**?

No!



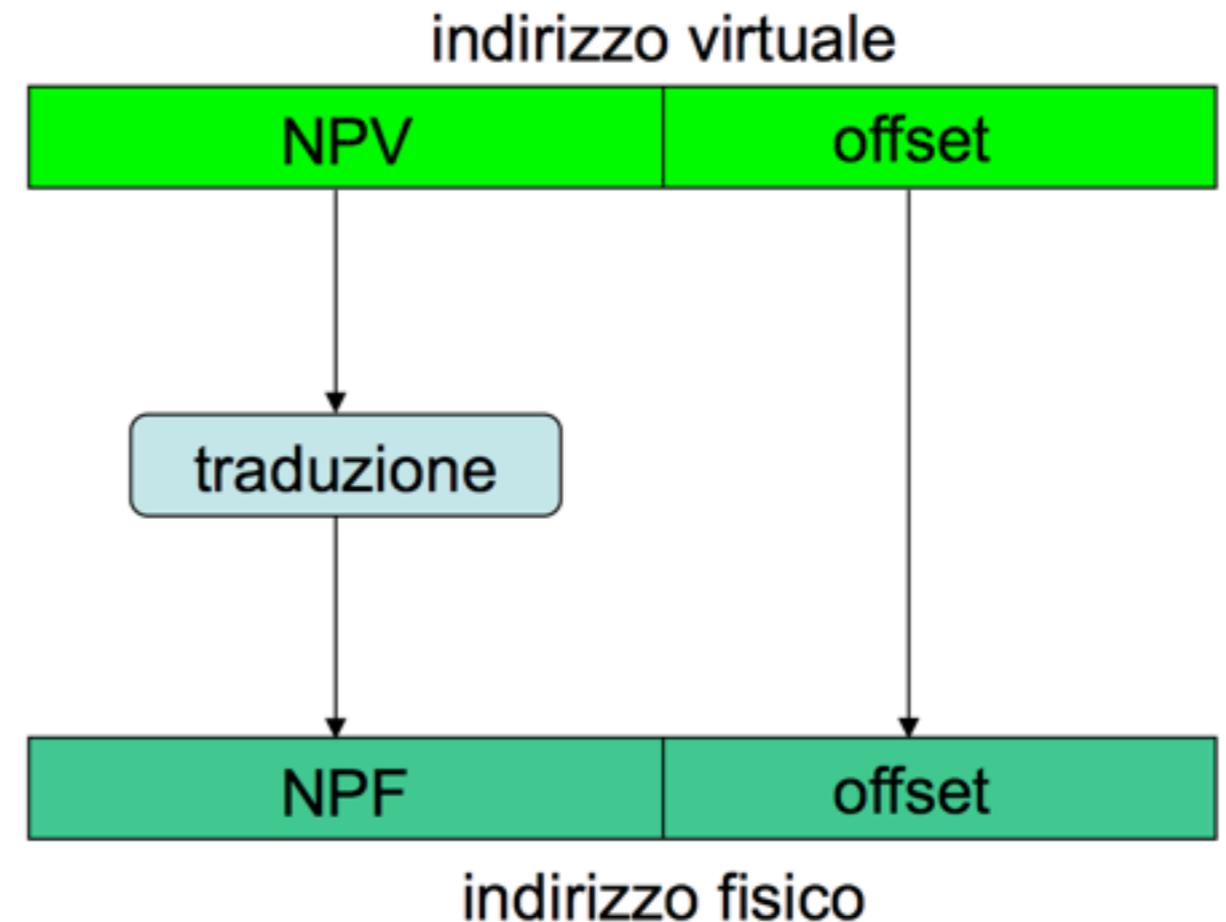
le pagine virtuali e quelle fisiche hanno la stessa dimensione, quindi l'offset è lo stesso

Memoria fisica e memoria virtuale (Es.3 - TE ?)

- Dimensione pagine 16 byte
-> **4 bit**
- L'indirizzo virtuale **00101000**
può corrispondere all'indirizzo
fisico **0010100**?

No!

Infatti, il primo indirizzo identifica il bit **1000** della pagina **0010** mentre il secondo identifica il bit **0100** della pagina **001**.



le pagine virtuali e quelle fisiche hanno la stessa dimensione, quindi l'offset è lo stesso

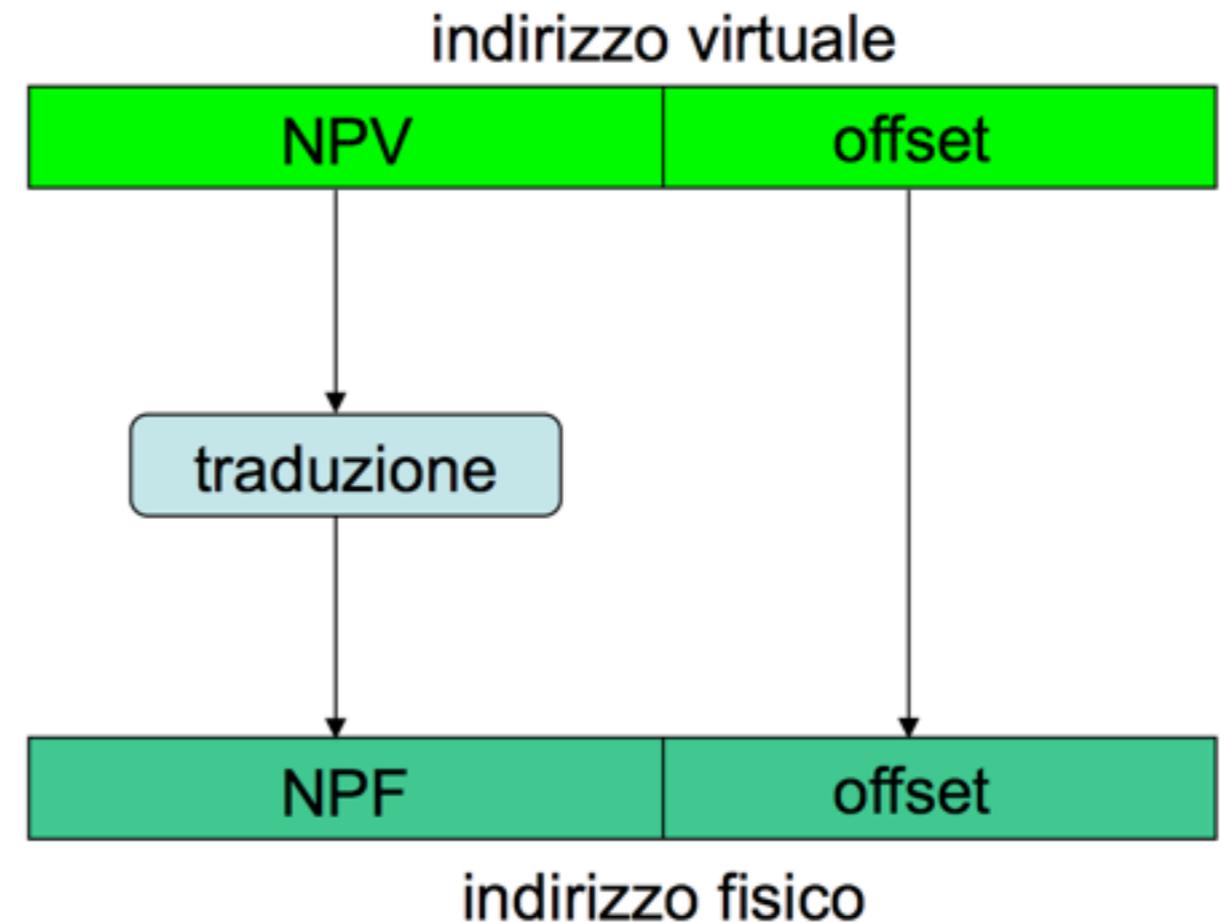
Memoria fisica e memoria virtuale (Es.3 - TE ?)

- Dimensione pagine 16 byte
-> **4 bit**
- L'indirizzo virtuale **00101000**
può corrispondere all'indirizzo
fisico **0010100**?

No!

Infatti, il primo indirizzo identifica il bit **1000** della pagina **0010** mentre il secondo identifica il bit **0100** della pagina **001**.

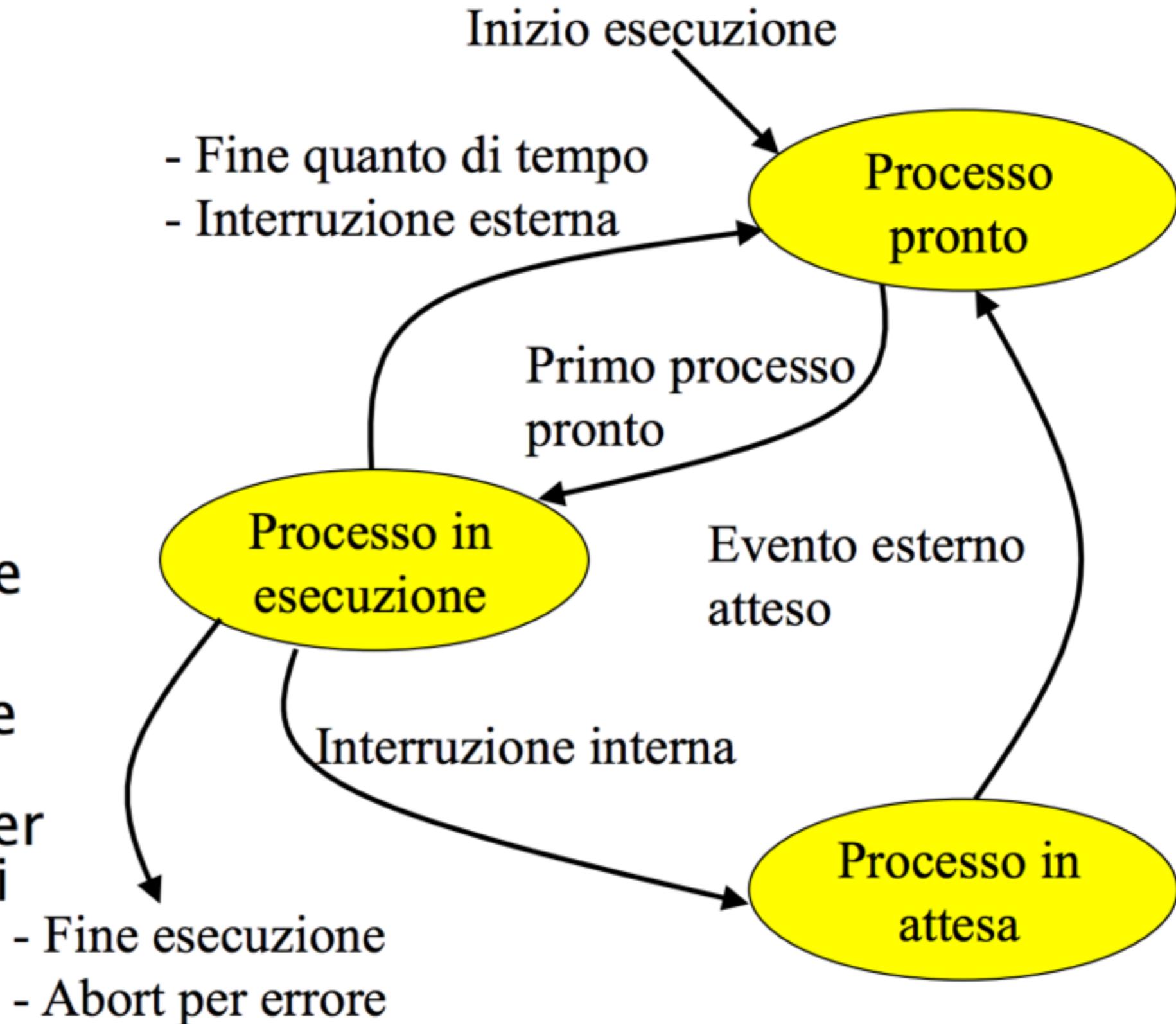
Siccome gli **spiazzamenti** dei bit all'interno delle pagine corrispondenti sono **sempre uguali** — cambiano infatti solo gli indici delle pagine — non abbiamo un match tra gli indirizzi.



le pagine virtuali e quelle fisiche hanno la stessa dimensione, quindi l'offset è lo stesso

Stato dei processi (teoria)

- **In esecuzione:** assegnato al processore ed eseguito da esso
- **Pronto:** può andare in esecuzione, se il gestore dei processi lo decide
- **In attesa:** attende il verificarsi di un evento esterno per andare in stato di pronto



Stato dei processi (Es.4 - TE 19/02/2015)

Si consideri un **sistema monoprocesso** in cui si avviano **quattro diversi processi** (A, B, C, D) che eseguono il seguente programma; si consideri ciascuna delle situazioni elencate qui sotto e si dica se può verificarsi.

```
#include <stdio.h>
int main() {
    int dato, risultato, i;

    while (1){
        risultato=0;
        printf("\n\nInserire un numero intero positivo da tastiera: ");
        scanf("%d", &dato);
        for (i=2; i<(dato/2)+1; i++)
            if (i*i==dato)
                risultato = i;
        printf("\nDato: %d\nRisultato: %d",dato, risultato);
    }
    return 0;
}
```

Stato dei processi (Es.4 - TE 19/02/2015)

1. A, B, C sono tutti e quattro in stato di pronto.

```
#include <stdio.h>
int main() {
    int dato, risultato, i;

    while (1){
        risultato=0;
        printf("\n\nInserire un numero intero positivo da tastiera: ");
        scanf("%d", &dato);
        for (i=2; i<(dato/2)+1; i++)
            if (i*i==dato)
                risultato = i;
        printf("\nDato: %d\nRisultato: %d",dato, risultato);
    }
    return 0;
}
```

Stato dei processi (Es.4 - TE 19/02/2015)

1. *A, B, C sono tutti e quattro in stato di pronto.*

Si – **tutti** gli eseguibili **caricati in memoria** ed associati al rispettivo processo, **in attesa** che lo **scheduler** li metta in esecuzione

```
#include <stdio.h>
int main() {
    int dato, risultato, i;

    while (1){
        risultato=0;
        printf("\n\nInserire un numero intero positivo da tastiera: ");
        scanf("%d", &dato);
        for (i=2; i<(dato/2)+1; i++)
            if (i*i==dato)
                risultato = i;
        printf("\nDato: %d\nRisultato: %d",dato, risultato);
    }
    return 0;
}
```

Stato dei processi (Es.4 - TE 19/02/2015)

2. A, B, C sono tutti e quattro in stato di attesa.

```
#include <stdio.h>
int main() {
    int dato, risultato, i;

    while (1){
        risultato=0;
        printf("\n\nInserire un numero intero positivo da tastiera: ");
        scanf("%d", &dato);
        for (i=2; i<(dato/2)+1; i++)
            if (i*i==dato)
                risultato = i;
        printf("\nDato: %d\nRisultato: %d",dato, risultato);
    }
    return 0;
}
```

Stato dei processi (Es.4 - TE 19/02/2015)

2. A, B, C sono tutti e quattro in stato di attesa.

```
#include <stdio.h>
int main() {
    int dato, risultato, i;

    while (1){
        risultato=0;
        printf("\n\nInserire un numero intero positivo da tastiera: ");
        scanf("%d", &dato);
        for (i=2; i<(dato/2)+1; i++)
            if (i*i==dato)
                risultato = i;
        printf("\nDato: %d\nRisultato: %d",dato, risultato);
    }
    return 0;
}
```

Stato dei processi (Es.4 - TE 19/02/2015)

2. A, B, C sono tutti e quattro in stato di attesa.

Si – ad esempio tutti **bloccati sulla scanf**

```
#include <stdio.h>
int main() {
    int dato, risultato, i;

    while (1){
        risultato=0;
        printf("\n\nInserire un numero intero positivo da tastiera: ");
        scanf("%d", &dato);
        for (i=2; i<(dato/2)+1; i++)
            if (i*i==dato)
                risultato = i;
        printf("\nDato: %d\nRisultato: %d",dato, risultato);
    }
    return 0;
}
```

Stato dei processi (Es.4 - TE 19/02/2015)

3. *A, B, sono in stato di esecuzione e C e D sono in stato di attesa.*

```
#include <stdio.h>
int main() {
    int dato, risultato, i;

    while (1){
        risultato=0;
        printf("\n\nInserire un numero intero positivo da tastiera: ");
        scanf("%d", &dato);
        for (i=2; i<(dato/2)+1; i++)
            if (i*i==dato)
                risultato = i;
        printf("\nDato: %d\nRisultato: %d",dato, risultato);
    }
    return 0;
}
```

Stato dei processi (Es.4 - TE 19/02/2015)

3. *A, B, sono in stato di esecuzione e C e D sono in stato di attesa.*

No – sistema **monoprocessore**

```
#include <stdio.h>
int main() {
    int dato, risultato, i;

    while (1){
        risultato=0;
        printf("\n\nInserire un numero intero positivo da tastiera: ");
        scanf("%d", &dato);
        for (i=2; i<(dato/2)+1; i++)
            if (i*i==dato)
                risultato = i;
        printf("\nDato: %d\nRisultato: %d",dato, risultato);
    }
    return 0;
}
```

Stato dei processi (Es.4 - TE 19/02/2015)

4. *A è in stato di attesa, B è in stato di esecuzione, C e D sono in stato di pronto.*

```
#include <stdio.h>
int main() {
    int dato, risultato, i;

    while (1){
        risultato=0;
        printf("\n\nInserire un numero intero positivo da tastiera: ");
        scanf("%d", &dato);
        for (i=2; i<(dato/2)+1; i++)
            if (i*i==dato)
                risultato = i;
        printf("\nDato: %d\nRisultato: %d",dato, risultato);
    }
    return 0;
}
```

Stato dei processi (Es.4 - TE 19/02/2015)

4. *A è in stato di attesa, B è in stato di esecuzione, C e D sono in stato di pronto.*

Si – ad esempio: **A è bloccata** sulla scanf, **B è in esecuzione** sul processore, **C e D sono pronti**, in attesa che lo scheduler li metta in esecuzione

```
#include <stdio.h>
int main() {
    int dato, risultato, i;

    while (1){
        risultato=0;
        printf("\n\nInserire un numero intero positivo da tastiera: ");
        scanf("%d", &dato);
        for (i=2; i<(dato/2)+1; i++)
            if (i*i==dato)
                risultato = i;
        printf("\nDato: %d\nRisultato: %d",dato, risultato);
    }
    return 0;
}
```

Ordine superiore (Es3 - TE 19/2/2015)

- Si consideri la seguente funzione di ordine superiore:

```
function r = fun(v, f)
    r = v(1);
    for i = 2:length(v)
        r = f(r, v(i));
    end
```

Ordine superiore (Es3 - TE 19/2/2015)

- Si consideri la seguente funzione di ordine superiore:

```
function r = fun(v,f)
    r = v(1);
    for i = 2:length(v)
        r = f(r,v(i));
    end
```

- A. Qual è il valore ritornato dalla chiamata **fun([8 9 10 9 6 0],@max)**?
Giustificare la risposta. `max` è una funzione offerta dalla libreria di Matlab. In particolare, se X e Y sono due scalari, `max(X,Y)` ritorna il maggiore tra i due valori.

Ordine superiore (Es3 - TE 19/2/2015)

- Si consideri la seguente funzione di ordine superiore:

```
function r = fun(v, f)
    r = v(1);
    for i = 2:length(v)
        r = f(r, v(i));
    end
```

- A. Qual è il valore ritornato dalla chiamata **fun([8 9 10 9 6 0], @max)**?
Giustificare la risposta. `max` è una funzione offerta dalla libreria di Matlab. In particolare, se X e Y sono due scalari, `max(X, Y)` ritorna il maggiore tra i due valori.
- B. Dire che cosa fa la funzione `fun` quando l'argomento v è un vettore numerico (di lunghezza pari almeno ad 1) ed f è `@max`

Ordine superiore (Es3 - TE 19/2/2015)

- Si consideri la seguente funzione di ordine superiore:

```
function r = fun(v, f)
    r = v(1);
    for i = 2:length(v)
        r = f(r, v(i));
    end
```

- A. Qual è il valore ritornato dalla chiamata **fun([8 9 10 9 6 0], @max)**?
Giustificare la risposta. `max` è una funzione offerta dalla libreria di Matlab. In particolare, se X e Y sono due scalari, `max(X, Y)` ritorna il maggiore tra i due valori.
- B. Dire che cosa fa la funzione `fun` quando l'argomento v è un vettore numerico (di lunghezza pari almeno ad 1) ed f è `@max`
- C. Implementare una versione ricorsiva della funzione `fun` che non richieda di utilizzare cicli (`for` o `while`). L'intestazione della funzione deve rimanere invariata.

Ordine superiore (Es3 - TE 19/2/2015)

A. Qual è il valore ritornato dalla chiamata **fun([8 9 10 9 6 0],@max)**?

```
function r = fun(v, f)
    r = v(1);
    for i = 2:length(v)
        r = f(r, v(i));
    end
```

Ordine superiore (Es3 - TE 19/2/2015)

A. Qual è il valore ritornato dalla chiamata **fun([8 9 10 9 6 0],@max)**?

```
function r = fun(v,f)           v = [8 9 10 9 6 0] , f = max(X,Y)
    r = v(1);
    for i = 2:length(v)
        r = f(r,v(i));
    end
```

Ordine superiore (Es3 - TE 19/2/2015)

A. Qual è il valore ritornato dalla chiamata **fun([8 9 10 9 6 0],@max)**?

```
function r = fun(v, f)
    r = v(1);
    for i = 2:length(v)
        r = f(r, v(i));
    end
```

$v = [8 \ 9 \ 10 \ 9 \ 6 \ 0]$, $f = \text{max}(X, Y)$

$r = 8$

Ordine superiore (Es3 - TE 19/2/2015)

A. Qual è il valore ritornato dalla chiamata **fun([8 9 10 9 6 0],@max)**?

```
function r = fun(v, f)           v = [8 9 10 9 6 0] , f = max(X,Y)
    r = v(1);                   r = 8
    for i = 2:length(v)        i = 2 , f(8, 9) = max(8,9)
        r = f(r, v(i));
    end
```

Ordine superiore (Es3 - TE 19/2/2015)

A. Qual è il valore ritornato dalla chiamata **fun([8 9 10 9 6 0],@max)**?

```
function r = fun(v, f)           v = [8 9 10 9 6 0] , f = max(X,Y)
    r = v(1);                   r = 8
    for i = 2:length(v)        i = 2 , f(8, 9) = max(8,9)      r = 9
        r = f(r, v(i));
    end
```

Ordine superiore (Es3 - TE 19/2/2015)

A. Qual è il valore ritornato dalla chiamata **fun([8 9 10 9 6 0],@max)**?

```
function r = fun(v, f)
    r = v(1);
    for i = 2:length(v)
        r = f(r, v(i));
    end
```

$v = [8 \ 9 \ 10 \ 9 \ 6 \ 0]$, $f = \text{max}(X, Y)$

$r = 8$

$i = 2$, $f(8, 9) = \text{max}(8, 9)$ $r = 9$

$i = 3$, $f(9, 10) = \text{max}(9, 10)$

Ordine superiore (Es3 - TE 19/2/2015)

A. Qual è il valore ritornato dalla chiamata **fun([8 9 10 9 6 0],@max)**?

```
function r = fun(v, f)
    r = v(1);
    for i = 2:length(v)
        r = f(r, v(i));
    end
```

$v = [8 \ 9 \ 10 \ 9 \ 6 \ 0]$, $f = \text{max}(X, Y)$

$r = 8$

$i = 2$, $f(8, 9) = \text{max}(8, 9)$ $r = 9$

$i = 3$, $f(9, 10) = \text{max}(9, 10)$ $r = 10$

Ordine superiore (Es3 - TE 19/2/2015)

A. Qual è il valore ritornato dalla chiamata **fun([8 9 10 9 6 0],@max)**?

```
function r = fun(v, f)
    r = v(1);
    for i = 2:length(v)
        r = f(r, v(i));
    end
```

$v = [8 \ 9 \ 10 \ 9 \ 6 \ 0]$, $f = \text{max}(X, Y)$

$r = 8$

$i = 2$, $f(8, 9) = \text{max}(8, 9)$ $r = 9$

$i = 3$, $f(9, 10) = \text{max}(9, 10)$ $r = 10$

$i = 4$, $f(10, 9) = \text{max}(10, 9)$

Ordine superiore (Es3 - TE 19/2/2015)

A. Qual è il valore ritornato dalla chiamata **fun([8 9 10 9 6 0],@max)**?

```
function r = fun(v, f)
    r = v(1);
    for i = 2:length(v)
        r = f(r, v(i));
    end
```

$v = [8 \ 9 \ 10 \ 9 \ 6 \ 0]$, $f = \text{max}(X, Y)$

$r = 8$

$i = 2$, $f(8, 9) = \text{max}(8, 9)$ $r = 9$

$i = 3$, $f(9, 10) = \text{max}(9, 10)$ $r = 10$

$i = 4$, $f(10, 9) = \text{max}(10, 9)$ $r = 10$

Ordine superiore (Es3 - TE 19/2/2015)

A. Qual è il valore ritornato dalla chiamata **fun([8 9 10 9 6 0],@max)**?

```
function r = fun(v, f)
    r = v(1);
    for i = 2:length(v)
        r = f(r, v(i));
    end
```

$v = [8 \ 9 \ 10 \ 9 \ 6 \ 0]$, $f = \text{max}(X, Y)$

$r = 8$

$i = 2$, $f(8, 9) = \text{max}(8, 9)$ $r = 9$

$i = 3$, $f(9, 10) = \text{max}(9, 10)$ $r = 10$

$i = 4$, $f(10, 9) = \text{max}(10, 9)$ $r = 10$

$i = 5$, $f(10, 6) = \text{max}(10, 6)$

Ordine superiore (Es3 - TE 19/2/2015)

A. Qual è il valore ritornato dalla chiamata **fun([8 9 10 9 6 0],@max)**?

```
function r = fun(v, f)
    r = v(1);
    for i = 2:length(v)
        r = f(r, v(i));
    end
```

$v = [8 \ 9 \ 10 \ 9 \ 6 \ 0]$, $f = \text{max}(X, Y)$

$r = 8$

$i = 2$, $f(8, 9) = \text{max}(8, 9)$ $r = 9$

$i = 3$, $f(9, 10) = \text{max}(9, 10)$ $r = 10$

$i = 4$, $f(10, 9) = \text{max}(10, 9)$ $r = 10$

$i = 5$, $f(10, 6) = \text{max}(10, 6)$ $r = 10$

Ordine superiore (Es3 - TE 19/2/2015)

A. Qual è il valore ritornato dalla chiamata **fun([8 9 10 9 6 0],@max)**?

```
function r = fun(v, f)
    r = v(1);
    for i = 2:length(v)
        r = f(r, v(i));
    end
```

$v = [8 \ 9 \ 10 \ 9 \ 6 \ 0]$, $f = \text{max}(X, Y)$

$r = 8$

$i = 2$, $f(8, 9) = \text{max}(8, 9)$ $r = 9$

$i = 3$, $f(9, 10) = \text{max}(9, 10)$ $r = 10$

$i = 4$, $f(10, 9) = \text{max}(10, 9)$ $r = 10$

$i = 5$, $f(10, 6) = \text{max}(10, 6)$ $r = 10$

$i = 6$, $f(10, 0) = \text{max}(10, 0)$

Ordine superiore (Es3 - TE 19/2/2015)

A. Qual è il valore ritornato dalla chiamata **fun([8 9 10 9 6 0],@max)**?

```
function r = fun(v, f)
    r = v(1);
    for i = 2:length(v)
        r = f(r, v(i));
    end
```

$v = [8 \ 9 \ 10 \ 9 \ 6 \ 0]$, $f = \text{max}(X, Y)$

$r = 8$

$i = 2$, $f(8, 9) = \text{max}(8, 9)$ $r = 9$

$i = 3$, $f(9, 10) = \text{max}(9, 10)$ $r = 10$

$i = 4$, $f(10, 9) = \text{max}(10, 9)$ $r = 10$

$i = 5$, $f(10, 6) = \text{max}(10, 6)$ $r = 10$

$i = 6$, $f(10, 0) = \text{max}(10, 0)$ **$r = 10$**

Ordine superiore (Es3 - TE 19/2/2015)

A. Qual è il valore ritornato dalla chiamata **fun([8 9 10 9 6 0],@max)**?

```
function r = fun(v,f)
    r = v(1);
    for i = 2:length(v)
        r = f(r,v(i));
    end
```

v = **[8 9 10 9 6 0]** , f = **max(X,Y)**

r = 8

i = 2 , f(**8**, **9**) = max(8,9) r = 9

i = 3 , f(9, **10**) = max(9,10) r = 10

i = 4 , f(10, **9**) = max(10,9) r = 10

i = 5 , f(10, **6**) = max(10,6) r = 10

i = 6 , f(10, **0**) = max(10,0) **r = 10**

B. Dire che cosa fa la funzione fun quando l'argomento v è un vettore numerico (di lunghezza pari almeno ad 1) ed f è @max

Ordine superiore (Es3 - TE 19/2/2015)

A. Qual è il valore ritornato dalla chiamata **fun([8 9 10 9 6 0],@max)**?

function r = fun(v, f)	v = [8 9 10 9 6 0] , f = max(X,Y)
r = v(1);	r = 8
for i = 2:length(v)	i = 2 , f(8 , 9) = max(8,9) r = 9
r = f(r, v(i));	i = 3 , f(9, 10) = max(9,10) r = 10
end	i = 4 , f(10, 9) = max(10,9) r = 10
	i = 5 , f(10, 6) = max(10,6) r = 10
	i = 6 , f(10, 0) = max(10,0) r = 10

B. Dire che cosa fa la funzione fun quando l'argomento v è un vettore numerico (di lunghezza pari almeno ad 1) ed f è @max

La funzione restituisce **l'elemento di maggior valore di v**

Ordine superiore (Es3 - TE 19/2/2015)

C. Implementare una versione ricorsiva della funzione fun che non richieda di utilizzare cicli (for o while). L'intestazione della funzione deve rimanere invariata.

```
function r = fun(v, f)
```

```
function r = fun(v, f)
    r = v(1);
    for i = 2:length(v)
        r = f(r, v(i));
    end
```

Ordine superiore (Es3 - TE 19/2/2015)

C. Implementare una versione ricorsiva della funzione fun che non richieda di utilizzare cicli (for o while). L'intestazione della funzione deve rimanere invariata.

```
function r = fun(v, f)
    n = length(v);
```

```
function r = fun(v, f)
    r = v(1);
    for i = 2:length(v)
        r = f(r, v(i));
    end
```

Ordine superiore (Es3 - TE 19/2/2015)

C. Implementare una versione ricorsiva della funzione fun che non richieda di utilizzare cicli (for o while). L'intestazione della funzione deve rimanere invariata.

```
function r = fun(v, f)
    n = length(v);
    if
    else
    end
```

```
function r = fun(v, f)
    r = v(1);
    for i = 2:length(v)
        r = f(r, v(i));
    end
```

Ordine superiore (Es3 - TE 19/2/2015)

C. Implementare una versione ricorsiva della funzione fun che non richieda di utilizzare cicli (for o while). L'intestazione della funzione deve rimanere invariata.

```
function r = fun(v, f)
    n = length(v);
    if

    else

    end
```

```
function r = fun(v, f)
    r = v(1);
    for i = 2:length(v)
        r = f(r, v(i));
    end
```

Ordine superiore (Es3 - TE 19/2/2015)

C. Implementare una versione ricorsiva della funzione fun che non richieda di utilizzare cicli (for o while). L'intestazione della funzione deve rimanere invariata.

```
function r = fun(v, f)
    n = length(v);
    if

    else

    end
```

```
function r = fun(v, f)
    r = v(1);
    for i = 2:length(v)
        r = f(r, v(i));
    end
```

Ordine superiore (Es3 - TE 19/2/2015)

C. Implementare una versione ricorsiva della funzione fun che non richieda di utilizzare cicli (for o while). L'intestazione della funzione deve rimanere invariata.

```
function r = fun(v, f)
    n = length(v);
    if
        n < 2
            r = v(1);
        else
            r = f(r, v(2:n));
        end
end
```

```
function r = fun(v, f)
    r = v(1);
    for i = 2:length(v)
        r = f(r, v(i));
    end
```

Ordine superiore (Es3 - TE 19/2/2015)

C. Implementare una versione ricorsiva della funzione fun che non richieda di utilizzare cicli (for o while). L'intestazione della funzione deve rimanere invariata.

```
function r = fun(v, f)
    n = length(v);
    if
        n < 2
            r = v(1);
        else
            r = f(v(1), fun(v(2:n), f));
        end
end
```

```
function r = fun(v, f)
    r = v(1);
    for i = 2:length(v)
        r = f(r, v(i));
    end
```

Ordine superiore (Es3 - TE 19/2/2015)

C. Implementare una versione ricorsiva della funzione fun che non richieda di utilizzare cicli (for o while). L'intestazione della funzione deve rimanere invariata.

```
function r = fun(v, f)
    n = length(v);
    if
        n < 2
            r = v(1);
        else
            r = f(v(1), fun(v(2:n), f));
        end
end
```

```
function r = fun(v, f)
    r = v(1);
    for i = 2:length(v)
        r = f(r, v(i));
    end
```

Ordine superiore (Es3 - TE 19/2/2015)

C. Implementare una versione ricorsiva della funzione fun che non richieda di utilizzare cicli (for o while). L'intestazione della funzione deve rimanere invariata.

```
function r = fun(v, f)
    n = length(v);
    if
```

```
else
```

```
    r = f(      , fun(v(2:end), f));
```

```
end
```

```
function r = fun(v, f)
    r = v(1);
    for i = 2:length(v)
        r = f(r, v(i));
    end
```

Ordine superiore (Es3 - TE 19/2/2015)

C. Implementare una versione ricorsiva della funzione fun che non richieda di utilizzare cicli (for o while). L'intestazione della funzione deve rimanere invariata.

```
function r = fun(v, f)
    n = length(v);
    if
```

```
else
```

```
    r = f(v(1), fun(v(2:end), f));
```

```
end
```

```
function r = fun(v, f)
    r = v(1);
    for i = 2:length(v)
        r = f(r, v(i));
    end
```

Ordine superiore (Es3 - TE 19/2/2015)

C. Implementare una versione ricorsiva della funzione fun che non richieda di utilizzare cicli (for o while). L'intestazione della funzione deve rimanere invariata.

```
function r = fun(v, f)
    n = length(v);
    if n == 1
        r = v;
    else
        r = f(v(1), fun(v(2:end), f));
    end
```

```
function r = fun(v, f)
    r = v(1);
    for i = 2:length(v)
        r = f(r, v(i));
    end
```

Ordine superiore (Es3 - TE 19/2/2015)

```
function r = fun(v, f)
    n = length(v);
    if n == 1
        r = v;
    else
        r = f(v(1), fun(v(2:end), f));
    end
```

Ordine superiore (Es3 - TE 19/2/2015)

```
function r = fun(v, f)
    n = length(v);
    if n == 1
        r = v;
    else
        r = f(v(1), fun(v(2:end), f));
    end
```

```
fun([8 9 10 9 6 0], f = max(X,Y))
```

Ordine superiore (Es3 - TE 19/2/2015)

```
function r = fun(v, f)
    n = length(v);
    if n == 1
        r = v;
    else
        r = f(v(1), fun(v(2:end), f));
    end
```

```
fun([8 9 10 9 6 0], f = max(X,Y))
```

```
...
r = max( 8 , fun([9 10 9 6 0], f = max(X,Y)))
```

Ordine superiore (Es3 - TE 19/2/2015)

```
function r = fun(v, f)
    n = length(v);
    if n == 1
        r = v;
    else
        r = f(v(1), fun(v(2:end), f));
    end
```

```
fun([8 9 10 9 6 0], f = max(X,Y))
```

```
    ...
    r = max( 8 , fun([9 10 9 6 0], f = max(X,Y)))
```

```
        ...
        r = max( 9 , fun([10 9 6 0], f = max(X,Y)))
```

Ordine superiore (Es3 - TE 19/2/2015)

```
function r = fun(v, f)
    n = length(v);
    if n == 1
        r = v;
    else
        r = f(v(1), fun(v(2:end), f));
    end
```

```
fun([8 9 10 9 6 0], f = max(X,Y))
```

```
    ...  
    r = max( 8 , fun([9 10 9 6 0] , f = max(X,Y)))
```

```
        ...  
        r = max( 9 , fun([10 9 6 0] , f = max(X,Y)))
```

```
            ...  
            r = max( 10 , fun([9 6 0] , f = max(X,Y)))
```

Ordine superiore (Es3 - TE 19/2/2015)

```
function r = fun(v, f)
    n = length(v);
    if n == 1
        r = v;
    else
        r = f(v(1), fun(v(2:end), f));
    end
```

```
fun([8 9 10 9 6 0], f = max(X,Y))
    ...
    r = max( 8, fun([9 10 9 6 0], f = max(X,Y)))
    ...
    r = max( 9, fun([10 9 6 0], f = max(X,Y)))
    ...
    r = max( 10, fun([9 6 0], f = max(X,Y)))
    ...
    r = max( 9, fun([6 0], f = max(X,Y)))
```

Ordine superiore (Es3 - TE 19/2/2015)

```
function r = fun(v, f)
    n = length(v);
    if n == 1
        r = v;
    else
        r = f(v(1), fun(v(2:end), f));
    end
```

```
fun([8 9 10 9 6 0], f = max(X,Y))
```

```
    ...
    r = max( 8 , fun([9 10 9 6 0], f = max(X,Y)))
```

```
    ...
    r = max( 9 , fun([10 9 6 0], f = max(X,Y)))
```

```
    ...
    r = max( 10 , fun([9 6 0], f = max(X,Y)))
```

```
    ...
    r = max( 9 , fun([6 0], f = max(X,Y)))
```

```
    ...
    r = max( 6 , fun([0], f = max(X,Y)))
```

Ordine superiore (Es3 - TE 19/2/2015)

```
function r = fun(v, f)
    n = length(v);
    if n == 1
        r = v;
    else
        r = f(v(1), fun(v(2:end), f));
    end
```

```
fun([8 9 10 9 6 0], f = max(X,Y))
```

```
    ...
    r = max( 8 , fun([9 10 9 6 0], f = max(X,Y)))
```

```
    ...
    r = max( 9 , fun([10 9 6 0], f = max(X,Y)))
```

```
    ...
    r = max( 10 , fun([9 6 0], f = max(X,Y)))
```

```
    ...
    r = max( 9 , fun([6 0], f = max(X,Y)))
```

```
    ...
    r = max( 6 , fun([0], f = max(X,Y)))
```

```
    r = 0
```

Ordine superiore (Es3 - TE 19/2/2015)

```
function r = fun(v, f)
    n = length(v);
    if n == 1
        r = v;
    else
        r = f(v(1), fun(v(2:end), f));
    end
```

fun(**[8 9 10 9 6 0]**, f = **max(X,Y)**)

...
r = **max**(**8** , fun(**[9 10 9 6 0]** , f = **max(X,Y)**))

...
r = **max**(**9** , fun(**[10 9 6 0]** , f = **max(X,Y)**))

...
r = **max**(**10** , fun(**[9 6 0]** , f = **max(X,Y)**))

...
r = **max**(**9** , fun(**[6 0]** , f = **max(X,Y)**))

...
r = **max**(**6** , fun(**[0]** , f = **max(X,Y)**))

r = **0**

Ordine superiore (Es3 - TE 19/2/2015)

```
function r = fun(v, f)
    n = length(v);
    if n == 1
        r = v;
    else
        r = f(v(1), fun(v(2:end), f));
    end
```

fun(**[8 9 10 9 6 0]**, f = **max(X,Y)**)

...
r = **max**(**8** , fun(**[9 10 9 6 0]** , f = **max(X,Y)**))

...
r = **max**(**9** , fun(**[10 9 6 0]** , f = **max(X,Y)**))

...
r = **max**(**10** , fun(**[9 6 0]** , f = **max(X,Y)**))

...
r = **max**(**9** , fun(**[6 0]** , f = **max(X,Y)**))

...
r = **max**(**6** , fun(**[0]** , f = **max(X,Y)**))

r = 0

Ordine superiore (Es3 - TE 19/2/2015)

```
function r = fun(v, f)
    n = length(v);
    if n == 1
        r = v;
    else
        r = f(v(1), fun(v(2:end), f));
    end
```

fun(**[8 9 10 9 6 0]**, f = max(X,Y))

...
r = **max**(**8** , fun(**[9 10 9 6 0]**, f = max(X,Y)))

...
r = **max**(**9** , fun(**[10 9 6 0]**, f = max(X,Y)))

...
r = **max**(**10** , fun(**[9 6 0]**, f = max(X,Y)))

...
r = **max**(**9** , fun(**[6 0]**, f = max(X,Y)))

...
r = **max**(**6** , fun(**[0]**, f = max(X,Y)))

r = 0

Ordine superiore (Es3 - TE 19/2/2015)

```
function r = fun(v, f)
    n = length(v);
    if n == 1
        r = v;
    else
        r = f(v(1), fun(v(2:end), f));
    end
```

```
fun([8 9 10 9 6 0], f = max(X,Y))
```

```
    ...
    r = max( 8 , fun([9 10 9 6 0], f = max(X,Y)))
```

```
        ...
        r = max( 9 , fun([10 9 6 0], f = max(X,Y)))
```

```
            ...
            r = max( 10 , fun([9 6 0], f = max(X,Y)))
```

```
                ...
                r = max( 9 , fun([6 0], f = max(X,Y)))
```

```
                    r = max( 6 , 0) = 6
```

Ordine superiore (Es3 - TE 19/2/2015)

```
function r = fun(v, f)
    n = length(v);
    if n == 1
        r = v;
    else
        r = f(v(1), fun(v(2:end), f));
    end
```

```
fun([8 9 10 9 6 0], f = max(X,Y))
```

```
    ...
    r = max( 8 , fun([9 10 9 6 0] , f = max(X,Y)))
```

```
    ...
    r = max( 9 , fun([10 9 6 0] , f = max(X,Y)))
```

```
    ...
    r = max( 10 , fun([9 6 0] , f = max(X,Y)))
```

```
    ...
    r = max( 9 , fun([6 0] , f = max(X,Y)))
```

```
    r = max( 6 , 0) = 6
```

Ordine superiore (Es3 - TE 19/2/2015)

```
function r = fun(v, f)
    n = length(v);
    if n == 1
        r = v;
    else
        r = f(v(1), fun(v(2:end), f));
    end
```

fun(**[8 9 10 9 6 0]**, f = **max(X,Y)**)

...
r = **max**(**8** , fun(**[9 10 9 6 0]** , f = **max(X,Y)**))

...
r = **max**(**9** , fun(**[10 9 6 0]** , f = **max(X,Y)**))

...
r = **max**(**10** , fun(**[9 6 0]** , f = **max(X,Y)**))

...
r = **max**(**9** , fun(**[6 0]** , f = **max(X,Y)**))

r = max(6 , 0) = 6

Ordine superiore (Es3 - TE 19/2/2015)

```
function r = fun(v, f)
    n = length(v);
    if n == 1
        r = v;
    else
        r = f(v(1), fun(v(2:end), f));
    end
```

```
fun([8 9 10 9 6 0], f = max(X,Y))
```

```
    ...
    r = max( 8 , fun([9 10 9 6 0], f = max(X,Y)))
```

```
        ...
        r = max( 9 , fun([10 9 6 0], f = max(X,Y)))
```

```
            ...
            r = max( 10 , fun([9 6 0], f = max(X,Y)))
```

```
                ...
                r = max( 9 , 6) = 9
```

Ordine superiore (Es3 - TE 19/2/2015)

```
function r = fun(v, f)
    n = length(v);
    if n == 1
        r = v;
    else
        r = f(v(1), fun(v(2:end), f));
    end
```

fun(**[8 9 10 9 6 0]**, f = **max(X,Y)**)

...
r = **max(8**, fun(**[9 10 9 6 0]**, f = **max(X,Y)**))

...
r = **max(9**, fun(**[10 9 6 0]**, f = **max(X,Y)**))

...
r = **max(10**, fun(**[9 6 0]**, f = **max(X,Y)**))

...
r = **max(9**, 6) = 9

Ordine superiore (Es3 - TE 19/2/2015)

```
function r = fun(v, f)
    n = length(v);
    if n == 1
        r = v;
    else
        r = f(v(1), fun(v(2:end), f));
    end
```

fun(**[8 9 10 9 6 0]**, f = **max(X,Y)**)

...
r = **max(8 , fun(**[9 10 9 6 0]**, f = **max(X,Y)**))**

...
r = **max(9 , fun(**[10 9 6 0]**, f = **max(X,Y)**))**

...
r = **max(10 , fun(**[9 6 0]**, f = **max(X,Y)**))**

...
r = max(9 , 6) = 9

Ordine superiore (Es3 - TE 19/2/2015)

```
function r = fun(v, f)
    n = length(v);
    if n == 1
        r = v;
    else
        r = f(v(1), fun(v(2:end), f));
    end
```

```
fun([8 9 10 9 6 0], f = max(X,Y))
```

```
    ...  
    r = max( 8 , fun([9 10 9 6 0], f = max(X,Y)))
```

```
        ...  
        r = max( 9 , fun([10 9 6 0], f = max(X,Y)))
```

```
            ...  
            r = max( 10 , 9) = 10
```

Ordine superiore (Es3 - TE 19/2/2015)

```
function r = fun(v, f)
    n = length(v);
    if n == 1
        r = v;
    else
        r = f(v(1), fun(v(2:end), f));
    end
```

```
fun([8 9 10 9 6 0], f = max(X,Y))
```

```
    ...
    r = max( 8 , fun([9 10 9 6 0] , f = max(X,Y)))
```

```
    ...
    r = max( 9 , fun([10 9 6 0] , f = max(X,Y)))
```

```
    ...
    r = max( 10 , 9) = 10
```

Ordine superiore (Es3 - TE 19/2/2015)

```
function r = fun(v, f)
    n = length(v);
    if n == 1
        r = v;
    else
        r = f(v(1), fun(v(2:end), f));
    end
```

```
fun([8 9 10 9 6 0], f = max(X,Y))
```

```
    ...
    r = max( 8 , fun([9 10 9 6 0] , f = max(X,Y)))
```

```
    ...
    r = max( 9 , fun([10 9 6 0] , f = max(X,Y)))
```

```
    ...
    r = max( 10 , 9) = 10
```

Ordine superiore (Es3 - TE 19/2/2015)

```
function r = fun(v, f)
    n = length(v);
    if n == 1
        r = v;
    else
        r = f(v(1), fun(v(2:end), f));
    end
```

```
fun([8 9 10 9 6 0], f = max(X,Y))
```

```
    ...
    r = max( 8 , fun([9 10 9 6 0], f = max(X,Y)))
```

```
        ...
        r = max( 9 , 10) = 10
```

Ordine superiore (Es3 - TE 19/2/2015)

```
function r = fun(v, f)
    n = length(v);
    if n == 1
        r = v;
    else
        r = f(v(1), fun(v(2:end), f));
    end
```

```
fun([8 9 10 9 6 0], f = max(X,Y))
```

```
...
r = max( 8 , fun([9 10 9 6 0], f = max(X,Y)))
```

```
...
r = max( 9 , 10) = 10
```

Ordine superiore (Es3 - TE 19/2/2015)

```
function r = fun(v, f)
    n = length(v);
    if n == 1
        r = v;
    else
        r = f(v(1), fun(v(2:end), f));
    end
```

fun(**[8 9 10 9 6 0]**, f = max(X,Y))

...
r = max(**8**, fun(**[9 10 9 6 0]**, f = max(X,Y)))

...
r = max(**9**, 10) = 10

Ordine superiore (Es3 - TE 19/2/2015)

```
function r = fun(v, f)
    n = length(v);
    if n == 1
        r = v;
    else
        r = f(v(1), fun(v(2:end), f));
    end
```

```
fun([8 9 10 9 6 0], f = max(X,Y))
    ...
    r = max( 8 , 10) = 10
```

Ordine superiore (Es3 - TE 19/2/2015)

```
function r = fun(v, f)
    n = length(v);
    if n == 1
        r = v;
    else
        r = f(v(1), fun(v(2:end), f));
    end
```

fun(**[8 9 10 9 6 0]**, f = **max(X,Y)**) 

...
r = **max(8 , 10) = 10**

Ordine superiore (Es3 - TE 19/2/2015)

```
function r = fun(v, f)
    n = length(v);
    if n == 1
        r = v;
    else
        r = f(v(1), fun(v(2:end), f));
    end
```

fun(**[8 9 10 9 6 0]**, f = **max(X,Y)**)  10

...
r = **max(8 , 10) = 10**

Iterazione vs. Ricorsione (Es.3 - TE 19/02/2015)

- Scrivere in MATLAB la funzione **ricorsiva** che, preso in ingresso un vettore P di K elementi interi ed un numero h (con $K < h$), restituisca al chiamante il vettore “*ruotato a destra*” di h posizioni.
- Si scriva quindi una seconda funzione che risolve lo stesso problema in maniera **iterativa**.

Iterazione vs. Ricorsione (Es.3 - TE 19/02/2015)

- Scrivere in MATLAB la funzione **ricorsiva** che, preso in ingresso un vettore P di K elementi interi ed un numero h (con $K < h$), restituisca al chiamante il vettore “*ruotato a destra*” di h posizioni.
- Si scriva quindi una seconda funzione che risolve lo stesso problema in maniera **iterativa**.

Esempio:

Input: $P = [1, 2, 3, 4, 5, 6]$, $h = 3$

Iterazione vs. Ricorsione (Es.3 - TE 19/02/2015)

- Scrivere in MATLAB la funzione **ricorsiva** che, preso in ingresso un vettore P di K elementi interi ed un numero h (con $K < h$), restituisca al chiamante il vettore “*ruotato a destra*” di h posizioni.
- Si scriva quindi una seconda funzione che risolve lo stesso problema in maniera **iterativa**.

Esempio:

Input: $P = [1, 2, 3, 4, 5, 6]$, $h = 3$



Iterazione vs. Ricorsione (Es.3 - TE 19/02/2015)

- Scrivere in MATLAB la funzione **ricorsiva** che, preso in ingresso un vettore P di K elementi interi ed un numero h (con $K < h$), restituisca al chiamante il vettore “*ruotato a destra*” di h posizioni.
- Si scriva quindi una seconda funzione che risolve lo stesso problema in maniera **iterativa**.

Esempio:

Input: $P = [1, 2, 3, 4, 5, 6]$, $h = 3$

Output: $P = [4, 5, 6, 1, 2, 3]$



Iterazione (Es.3 - TE 19/02/2015)

```
function out = g(P, h)
```

```
    for
```

```
    end
```

Iterazione (Es.3 - TE 19/02/2015)

```
function out = g(P, h)
```

```
    for H=1:h
```

```
    end
```

Iterazione (Es.3 - TE 19/02/2015)

```
function out = g(P, h)
```

```
    for H=1:h
```

```
        P = [P(end) P(1:end-1)];
```

```
    end
```

Iterazione (Es.3 - TE 19/02/2015)

```
function out = g(P, h)

    for H=1:h
        P = [P(end) P(1:end-1)];
    end
    out = P;
end
```

Iterazione (Es.3 - TE 19/02/2015)

```
function out = g(P, h)
    if (h<1)
        return
    end
    for H=1:h
        P = [P(end) P(1:end-1)];
    end
    out = P;
end
```

Iterazione (Es.3 - TE 19/02/2015)

```
function out = g(P, h)
    if (h<1)
        return
    end
    for H=1:h
        P = [P(end) P(1:end-1)];
    end
    out = P;
end
```

H = 0, P = [1, 2, 3, 4, 5, 6]

Iterazione (Es.3 - TE 19/02/2015)

```
function out = g(P, h)
    if (h<1)
        return
    end
    for H=1:h
        P = [P(end) P(1:end-1)];
    end
    out = P;
end
```

H = 0, P = [1, 2, 3, 4, 5, 6]

Iterazione (Es.3 - TE 19/02/2015)

```
function out = g(P, h)
    if (h<1)
        return
    end
    for H=1:h
        P = [P(end) P(1:end-1)];
    end
    out = P;
end
```

$H = 0, P = [1, 2, 3, 4, 5, 6]$

Iterazione (Es.3 - TE 19/02/2015)

```
function out = g(P, h)
    if (h<1)
        return
    end
    for H=1:h
        P = [P(end) P(1:end-1)];
    end
    out = P;
end
```

$H = 0, P = [1, 2, 3, 4, 5, 6]$

$H = 1, P = [6, 1, 2, 3, 4, 5]$

Iterazione (Es.3 - TE 19/02/2015)

```
function out = g(P, h)
    if (h<1)
        return
    end
    for H=1:h
        P = [P(end) P(1:end-1)];
    end
    out = P;
end
```

H = 0, P = [1, 2, 3, 4, 5, 6]

H = 1, P = [6, 1, 2, 3, 4, 5]

H = 2, P = [5, 6, 1, 2, 3, 4]

Iterazione (Es.3 - TE 19/02/2015)

```
function out = g(P, h)
    if (h<1)
        return
    end
    for H=1:h
        P = [P(end) P(1:end-1)];
    end
    out = P;
end
```

$H = 0, P = [1, 2, 3, 4, 5, 6]$

$H = 1, P = [6, 1, 2, 3, 4, 5]$

$H = 2, P = [5, 6, 1, 2, 3, 4]$

$H = 3, P = [4, 5, 6, 1, 2, 3]$

Ricorsione (Es.3 - TE 19/02/2015)

```
function out = rec(P, h)
```

Ricorsione (Es.3 - TE 19/02/2015)

```
function out = rec(P, h)
    if
    else
end
```

Ricorsione (Es.3 - TE 19/02/2015)

```
function out = rec(P, h)
    if
    else
        out = rec([P(end) P(1:end-1)], h-1);
    end
```

Ricorsione (Es.3 - TE 19/02/2015)

```
function out = rec(P, h)
    if
    else
        out = rec([P(end) P(1:end-1)], h-1);
    end
```

Ricorsione (Es.3 - TE 19/02/2015)

```
function out = rec(P, h)
    if
    else
        out = rec([P(end) P(1:end-1)], h-1);
    end
```

Ricorsione (Es.3 - TE 19/02/2015)

```
function out = rec(P, h)
    if
    else
        out = rec([P(end) P(1:end-1)], h-1);
    end
```

Ricorsione (Es.3 - TE 19/02/2015)

```
function out = rec(P, h)
    if (h==0)

    else
        out = rec([P(end) P(1:end-1)], h-1);
    end
end
```

Ricorsione (Es.3 - TE 19/02/2015)

```
function out = rec(P, h)
    if (h==0)
        out = P;
    else
        out = rec([P(end) P(1:end-1)], h-1);
    end
```

Ricorsione (Es.3 - TE 19/02/2015)

```
function out = rec(P, h)
    if (h==0)
        out = P;
    else
        out = rec([P(end) P(1:end-1)], h-1);
    end
```

```
rec([1, 2, 3, 4, 5, 6], 3)
```

Ricorsione (Es.3 - TE 19/02/2015)

```
function out = rec(P, h)
    if (h==0)
        out = P;
    else
        out = rec([P(end) P(1:end-1)], h-1);
    end
```

```
rec([1, 2, 3, 4, 5, 6], 3)
...
out = rec([6, 1, 2, 3, 4, 5], 2)
```

Ricorsione (Es.3 - TE 19/02/2015)

```
function out = rec(P, h)
    if (h==0)
        out = P;
    else
        out = rec([P(end) P(1:end-1)], h-1);
    end
```

rec([1, 2, 3, 4, 5, 6], 3)

...
out = rec([6, 1, 2, 3, 4, 5], 2)

Ricorsione (Es.3 - TE 19/02/2015)

```
function out = rec(P, h)
    if (h==0)
        out = P;
    else
        out = rec([P(end) P(1:end-1)], h-1);
    end
```

rec([1, 2, 3, 4, 5, 6], 3)

...
out = rec([6, 1, 2, 3, 4, 5], 2)

...
out = rec([5, 6, 1, 2, 3, 4], 1)

Ricorsione (Es.3 - TE 19/02/2015)

```
function out = rec(P, h)
    if (h==0)
        out = P;
    else
        out = rec([P(end) P(1:end-1)], h-1);
    end
```

rec([1, 2, 3, 4, 5, 6], 3)

...
out = rec([6, 1, 2, 3, 4, 5], 2)

...
out = rec([5, 6, 1, 2, 3, 4], 1)

Ricorsione (Es.3 - TE 19/02/2015)

```
function out = rec(P, h)
    if (h==0)
        out = P;
    else
        out = rec([P(end) P(1:end-1)], h-1);
    end
```

rec([1, 2, 3, 4, 5, 6], 3)

...
out = rec([6, 1, 2, 3, 4, 5], 2)

...
out = rec([5, 6, 1, 2, 3, 4], 1)

...
out = rec([4, 5, 6, 1, 2, 3], 0)

Ricorsione (Es.3 - TE 19/02/2015)

```
function out = rec(P, h)
    if (h==0)
        out = P;
    else
        out = rec([P(end) P(1:end-1)], h-1);
    end
```

rec([1, 2, 3, 4, 5, 6], 3)

...
out = rec([6, 1, 2, 3, 4, 5], 2)

...
out = rec([5, 6, 1, 2, 3, 4], 1)

...
out = rec([4, 5, 6, 1, 2, 3], 0)

Ricorsione (Es.3 - TE 19/02/2015)

```
function out = rec(P, h)
    if (h==0)
        out = P;
    else
        out = rec([P(end) P(1:end-1)], h-1);
    end
```

rec([1, 2, 3, 4, 5, 6], 3)

...
out = rec([6, 1, 2, 3, 4, 5], 2)

...
out = rec([5, 6, 1, 2, 3, 4], 1)

...
out = rec([4, 5, 6, 1, 2, 3], 0)

...
out = [4, 5, 6, 1, 2, 3]

Ricorsione (Es.3 - TE 19/02/2015)

```
function out = rec(P, h)
    if (h==0)
        out = P;
    else
        out = rec([P(end) P(1:end-1)], h-1);
    end
```

rec([1, 2, 3, 4, 5, 6], 3)

...

out = rec([6, 1, 2, 3, 4, 5], 2)

...

out = rec([5, 6, 1, 2, 3, 4], 1)

...

out = **[4, 5, 6, 1, 2, 3]** ←

...

out = [4, 5, 6, 1, 2, 3]

Ricorsione (Es.3 - TE 19/02/2015)

```
function out = rec(P, h)
    if (h==0)
        out = P;
    else
        out = rec([P(end) P(1:end-1)], h-1);
    end
```

rec([1, 2, 3, 4, 5, 6], 3)

...

out = rec([6, 1, 2, 3, 4, 5], 2)

...

out = **[4, 5, 6, 1, 2, 3]** ←

...

out = [4, 5, 6, 1, 2, 3]

...

out = [4, 5, 6, 1, 2, 3]

Ricorsione (Es.3 - TE 19/02/2015)

```
function out = rec(P, h)
    if (h==0)
        out = P;
    else
        out = rec([P(end) P(1:end-1)], h-1);
    end
```

rec([1, 2, 3, 4, 5, 6], 3)

...

out = **[4, 5, 6, 1, 2, 3]** ←

...

out = [4, 5, 6, 1, 2, 3]

...

out = [4, 5, 6, 1, 2, 3]

...

out = [4, 5, 6, 1, 2, 3]

Ricorsione (Es.3 - TE 19/02/2015)

```
function out = rec(P, h)
    if (h==0)
        out = P;
    else
        out = rec([P(end) P(1:end-1)], h-1);
    end
```

rec([1, 2, 3, 4, 5, 6], 3) \longrightarrow [4, 5, 6, 1, 2, 3]

...
out = [4, 5, 6, 1, 2, 3]

...
out = [4, 5, 6, 1, 2, 3]

...
out = [4, 5, 6, 1, 2, 3]

...
out = [4, 5, 6, 1, 2, 3]

Conggettura di Goldbach (Es.1 - TE 19/02/2020)

Scorsa
esercitazione

- In matematica, la **congettura di Goldbach** è uno dei più vecchi problemi irrisolti nella teoria dei numeri. Essa afferma che *ogni numero pari maggiore di 2 può essere scritto come somma di due numeri primi* (che possono essere anche uguali).

- In matematica, la **congettura di Goldbach** è uno dei più vecchi problemi irrisolti nella teoria dei numeri. Essa afferma che *ogni numero pari maggiore di 2 può essere scritto come somma di due numeri primi* (che possono essere anche uguali).
- Si scriva quindi una funzione in MATLAB (chiamata **goldbach**) che, dato un numero pari, stampa a video i due numeri primi che sommati lo compongono. Nel realizzare la funzione goldbach, si consiglia di **utilizzare una seconda funzione**, chiamata **primo**, che dato un numero x in ingresso dica se tale numero sia primo o meno. L'interfaccia della funzione primo è la seguente:

Conggettura di Goldbach (Es.1 - TE 19/02/2024)

Scorsa
esercitazione

- In matematica, la **congettura di Goldbach** è uno dei più vecchi problemi irrisolti nella teoria dei numeri. Essa afferma che *ogni numero pari maggiore di 2 può essere scritto come somma di due numeri primi* (che possono essere anche uguali).
- Si scriva quindi una funzione in MATLAB (chiamata **goldbach**) che, dato un numero pari, stampa a video i due numeri primi che sommati lo compongono. Nel realizzare la funzione goldbach, si consiglia di **utilizzare una seconda funzione**, chiamata **primo**, che dato un numero x in ingresso dica se tale numero sia primo o meno. L'interfaccia della funzione primo è la seguente:

```
function [ris] = primo(x)
```

Conggettura di Goldbach (Es.1 - TE 19/02/2024)

Scorsa
esercitazione

- In matematica, la **congettura di Goldbach** è uno dei più vecchi problemi irrisolti nella teoria dei numeri. Essa afferma che *ogni numero pari maggiore di 2 può essere scritto come somma di due numeri primi* (che possono essere anche uguali).
- Si scriva quindi una funzione in MATLAB (chiamata **goldbach**) che, dato un numero pari, stampa a video i due numeri primi che sommati lo compongono. Nel realizzare la funzione goldbach, si consiglia di **utilizzare una seconda funzione**, chiamata **primo**, che dato un numero x in ingresso dica se tale numero sia primo o meno. L'interfaccia della funzione primo è la seguente:

```
function [ris] = primo(x)
```

- Nota: questo esercizio richiede quindi la creazione di UNA SOLA funzione: goldbach

Conggettura di Goldbach (Es.1 - TE 19/02/2020)

```
function [x,y] = goldbach(n, primo)
```

Scorsa
esercitazione

Conggettura di Goldbach (Es.1 - TE 19/02/2020)

Scorsa
esercitazione

```
function [x,y] = goldbach(n, primo)

    if (mod(n,2) ~= 0 || n<=2)
        return
    end
```

Conggettura di Goldbach (Es.1 - TE 19/02/2020)

Scorsa
esercitazione

```
function [x,y] = goldbach(n, primo)
```

```
    if (mod(n,2) ~= 0 || n<=2)
```

```
        return
```

```
    end
```

```
        if ((x+y) == n)
```

```
            display([num2str(x) ' + ' num2str(y)])
```

```
            return
```

```
        end
```

Conggettura di Goldbach (Es.1 - TE 19/02/2020)

Scorsa
esercitazione

```
function [x,y] = goldbach(n, primo)
```

```
    if (mod(n,2) ~= 0 || n<=2)
```

```
        return
```

```
    end
```

```
        if (primo(y))
```

```
            if ((x+y) == n)
```

```
                display([num2str(x) ' + ' num2str(y)])
```

```
                return
```

```
            end
```

```
        end
```

Conggettura di Goldbach (Es.1 - TE 19/02/2020)

```
function [x,y] = goldbach(n, primo)

    if (mod(n,2) ~= 0 || n<=2)
        return
    end

    while true
        if (primo(x))

            if (primo(y))
                if ((x+y) == n)
                    display([num2str(x) ' + ' num2str(y)])
                    return
                end
            end
            x = x + 1
        end
        y = n - x
    end

end
```

Conggettura di Goldbach (Es.1 - TE 19/02/2019)

Scorsa
esercitazione

```
function [x,y] = goldbach(n, primo)

    if (mod(n,2) ~= 0 || n<=2)
        return
    end

    for x=2:n
        if (primo(x))

            if (primo(y))
                if ((x+y) == n)
                    display([num2str(x) ' + ' num2str(y)])
                    return
                end
            end
        end
    end
end
```

Conggettura di Goldbach (Es.1 - TE 19/02/2020)

```
function [x,y] = goldbach(n, primo)

    if (mod(n,2) ~= 0 || n<=2)
        return
    end

    for x=2:n
        if (primo(x))
            for y=2:x
                if (primo(y))
                    if ((x+y) == n)
                        display([num2str(x) ' + ' num2str(y)])
                        return
                    end
                end
            end
        end
    end
end
```

Conggettura di Goldbach (Es.1 - TE 19/02/2019)

Scorsa
esercitazione

```
function [x,y] = goldbach(n, primo)

    if (mod(n,2) ~= 0 || n<=2)
        return
    end

    for x=2:n
        if (primo(x))
            for y=2:x
                if (primo(y))
                    if ((x+y) == n)
                        display([num2str(x) ' + ' num2str(y)])
                        return
                    end
                end
            end
        end
    end

    display 'congettura errata *_*'
```

Esempio di uso di *goldbach*

```
% Esempio di invocazione goldbach  
numero = 936;
```

```
goldbach(numero, primo);
```

```
goldbach(numero, primo);
```

```
goldbach(numero, primo);
```

Esempio di uso di *goldbach*

```
% Esempio di invocazione goldbach
```

```
numero = 936;
```

```
primo = @(x) numeroPrimoIterativa(x);  
goldbach(numero, primo);
```

```
goldbach(numero, primo);
```

```
goldbach(numero, primo);
```

Esempio di uso di *goldbach*

```
% Esempio di invocazione goldbach
numero = 936;

primo = @(x) numeroPrimoIterativa(x);
goldbach(numero, primo);

primo = @(x) numeroPrimoRicorsiva(x,2);
goldbach(numero, primo);

goldbach(numero, primo);
```

Esempio di uso di *goldbach*

```
% Esempio di invocazione goldbach
numero = 936;

primo = @(x) numeroPrimoIterativa(x);
goldbach(numero, primo);

primo = @(x) numeroPrimoRicorsiva(x,2);
goldbach(numero, primo);

primo = @(x) numeroPrimoVettoriale(x);
goldbach(numero, primo);
```










Grazie

