



 POLITECNICO DI MILANO

Dipartimento di
Elettronica e Informazione

Simulazione seconda PI - Soluzioni

Matteo Ferroni
matteo.ferroni@polimi.it

24/01/2017



POLITECNICO
DI MILANO



Esercizio 1 - Soluzione

```
% Carico il contenuto del file in memoria  
load prod.mat;
```

Esercizio 1 - Soluzione

```
% Carico il contenuto del file in memoria
```

```
load prod.mat;
```

```
% Calcolo la media e lo stipendio
```

```
for ii = 1:length(archivio)
```

```
end
```

Esercizio 1 - Soluzione

```
% Carico il contenuto del file in memoria
load prod.mat;

% Calcolo la media e lo stipendio
for ii = 1:length(archivio)
    archivio(ii).media = mean(archivio(ii).ore);
```

```
end
```

Esercizio 1 - Soluzione

```
% Carico il contenuto del file in memoria
load prod.mat;

% Calcolo la media e lo stipendio
for ii = 1:length(archivio)
    archivio(ii).media = mean(archivio(ii).ore);

    if archivio(ii).media < 7
        % se un dipendente ha lavorato in media meno di 7 ore al giorno
        % la paga è di 5? all'ora
        euro_ora = 5;
    end
end
```

Esercizio 1 - Soluzione

```
% Carico il contenuto del file in memoria
load prod.mat;

% Calcolo la media e lo stipendio
for ii = 1:length(archivio)
    archivio(ii).media = mean(archivio(ii).ore);

    if archivio(ii).media < 7
        % se un dipendente ha lavorato in media meno di 7 ore al giorno
        % la paga è di 5? all'ora
        euro_ora = 5;
    elseif archivio(ii).media < 12
        % se ha lavorato da 7 a 12 ore in media, è di 10? all'ora;
        euro_ora = 10;
    end
end
```

Esercizio 1 - Soluzione

```
% Carico il contenuto del file in memoria
load prod.mat;

% Calcolo la media e lo stipendio
for ii = 1:length(archivio)
    archivio(ii).media = mean(archivio(ii).ore);

    if archivio(ii).media < 7
        % se un dipendente ha lavorato in media meno di 7 ore al giorno
        % la paga è di 5? all'ora
        euro_ora = 5;
    elseif archivio(ii).media < 12
        % se ha lavorato da 7 a 12 ore in media, è di 10? all'ora;
        euro_ora = 10;
    else
        % altrimenti è di 6? all'ora;
        euro_ora = 6;
    end
end
```

end

Esercizio 1 - Soluzione

```
% Carico il contenuto del file in memoria
load prod.mat;

% Calcolo la media e lo stipendio
for ii = 1:length(archivio)
    archivio(ii).media = mean(archivio(ii).ore);

    if archivio(ii).media < 7
        % se un dipendente ha lavorato in media meno di 7 ore al giorno
        % la paga è di 5? all'ora
        euro_ora = 5;
    elseif archivio(ii).media < 12
        % se ha lavorato da 7 a 12 ore in media, è di 10? all'ora;
        euro_ora = 10;
    else
        % altrimenti è di 6? all'ora;
        euro_ora = 6;
    end

    archivio(ii).stipendio = sum(archivio(ii).ore.*euro_ora);
end
```

end

Esercizio 1 - Soluzione

```
% Carico il contenuto del file in memoria
load prod.mat;

% Calcolo la media e lo stipendio
for ii = 1:length(archivio)
    archivio(ii).media = mean(archivio(ii).ore);

    if archivio(ii).media < 7
        % se un dipendente ha lavorato in media meno di 7 ore al giorno
        % la paga è di 5? all'ora
        euro_ora = 5;
    elseif archivio(ii).media < 12
        % se ha lavorato da 7 a 12 ore in media, è di 10? all'ora;
        euro_ora = 10;
    else
        % altrimenti è di 6? all'ora;
        euro_ora = 6;
    end

    archivio(ii).stipendio = sum(archivio(ii).ore.*euro_ora);

    if archivio(ii).contratti >= 48
        archivio(ii).stipendio = archivio(ii).stipendio + 500;
    end
end
end
```

Esercizio 1 - Soluzione

```
% stampi a video il numero dei dipendenti che, pur lavorando più di 12 ore  
% al giorno (in media), guadagnano meno della media dei dipendenti;
```

Esercizio 1 - Soluzione

```
% stampi a video il numero dei dipendenti che, pur lavorando più di 12 ore  
% al giorno (in media), guadagnano meno della media dei dipendenti;  
media_stipendi = mean([archivio.stipendio]);
```

Esercizio 1 - Soluzione

```
% stampi a video il numero dei dipendenti che, pur lavorando più di 12 ore  
% al giorno (in media), guadagnano meno della media dei dipendenti;  
media_stipendi = mean([archivio.stipendio]);  
num_dipendenti_sfortunati = sum([archivio.stipendio]<media_stipendi  
                                & [archivio.media]>12);
```

Esercizio 1 - Soluzione

```
% stampi a video il numero dei dipendenti che, pur lavorando più di 12 ore
% al giorno (in media), guadagnano meno della media dei dipendenti;
media_stipendi = mean([archivio.stipendio]);
num_dipendenti_sfortunati = sum([archivio.stipendio]<media_stipendi
                                & [archivio.media]>12);
disp(['Numero dipendenti sfortunati: ' num2str(num_dipendenti_sfortunati)]);

% dica se è vero che le donne al di sotto dei 35 anni sono in grado di
% stipulare in media più contratti rispetto agli uomini che hanno più di 35 a
```

Esercizio 1 - Soluzione

```
% stampi a video il numero dei dipendenti che, pur lavorando più di 12 ore
% al giorno (in media), guadagnano meno della media dei dipendenti;
media_stipendi = mean([archivio.stipendio]);
num_dipendenti_sfortunati = sum([archivio.stipendio]<media_stipendi
                                & [archivio.media]>12);
disp(['Numero dipendenti sfortunati: ' num2str(num_dipendenti_sfortunati)]);

% dica se è vero che le donne al di sotto dei 35 anni sono in grado di
% stipulare in media più contratti rispetto agli uomini che hanno più di 35
media_contratti_F_under_35 = mean([archivio([archivio.sesso]=='F'
                                             & [archivio.eta]<35).contratti]);
media_contratti_M_over_50 = mean([archivio([archivio.sesso]=='M'
                                             & [archivio.eta]>50).contratti]);
```

Esercizio 1 - Soluzione

```
% stampi a video il numero dei dipendenti che, pur lavorando più di 12 ore
% al giorno (in media), guadagnano meno della media dei dipendenti;
media_stipendi = mean([archivio.stipendio]);
num_dipendenti_sfortunati = sum([archivio.stipendio]<media_stipendi
                                & [archivio.media]>12);
disp(['Numero dipendenti sfortunati: ' num2str(num_dipendenti_sfortunati)]);

% dica se è vero che le donne al di sotto dei 35 anni sono in grado di
% stipulare in media più contratti rispetto agli uomini che hanno più di 35
media_contratti_F_under_35 = mean([archivio([archivio.sesso]=='F'
                                             & [archivio.eta]<35).contratti]);
media_contratti_M_over_50 = mean([archivio([archivio.sesso]=='M'
                                             & [archivio.eta]>50).contratti]);

if media_contratti_F_under_35 > media_contratti_M_over_50
    disp('Le donne al di sotto dei 35 anni vincono!');
else
    disp('Gli uomini che hanno più di 35 anni vincono!');
end
```

Esercizio 1 - Soluzione

```
% stampi gli stipendi medi delle donne e degli uomini che hanno più di 50 anni.
```

Esercizio 1 - Soluzione

```
% stampi gli stipendi medi delle donne e degli uomini che hanno più di 50 anni.  
stipendio_medio_F_over_50 = mean([archivio([archivio.sesso]=='F'  
                                         & [archivio.eta]>50).stipendio]);  
stipendio_medio_M_over_50 = mean([archivio([archivio.sesso]=='M'  
                                         & [archivio.eta]>50).stipendio]);
```

Esercizio 1 - Soluzione

```
% stampi gli stipendi medi delle donne e degli uomini che hanno più di 50 anni.
stipendio_medio_F_over_50 = mean([archivio([archivio.sesso]=='F'  
                                         & [archivio.eta]>50).stipendio]);
stipendio_medio_M_over_50 = mean([archivio([archivio.sesso]=='M'  
                                         & [archivio.eta]>50).stipendio]);

disp(['Le donne over 50 guadagnano, in media: '  
      num2str(stipendio_medio_F_over_50) ' euro al mese']);
disp(['Gli uomini over 50 guadagnano, in media: '  
      num2str(stipendio_medio_M_over_50) ' euro al mese']);
```

Esercizio 1 - Esempio creazione archivio

```
% Semplice script per la generazione di un archivio d'esempio
for ii = 1:1200
    archivio(ii).nome = ['Utente ' num2str(ii)];
    archivio(ii).eta = ceil(rand * 65);

    if rand > 0.5
        archivio(ii).sesso = 'M';
    else
        archivio(ii).sesso = 'F';
    end

    test = rand;
    if test < 0.3
        archivio(ii).ore = [6 7 6 7 5 6 7 6 7 5 6 7 6 7 5 6 7 6 7 5];
    elseif test < 0.6
        archivio(ii).ore = [8 9 8 7 8 8 9 8 7 8 8 9 8 7 8 8 9 8 7 8];
    elseif test < 0.8
        archivio(ii).ore = [11 12 11 10 12 11 12 11 10 12 11 12 11 10 12 11 12 11 10 12];
    else
        archivio(ii).ore = [13 14 15 14 13 13 14 15 14 13 13 14 15 14 13 13 14 15 14 13];
    end

    test = rand;
    if test < 0.3
        archivio(ii).contratti == 10
    elseif test < 0.6
        archivio(ii).contratti == 25
    elseif test < 0.9
        archivio(ii).contratti == 48
    else
        archivio(ii).contratti == 55
    end
end
end
```

Esercizio 2 - Esempi

esempio esito positivo:

$V1 = [a\ c\ d\ e\ b]$ $V2 = [b\ e\ a]$

esempio esito negativo:

$V1 = [a\ c\ d\ e\ b]$ $V2 = [e\ f\ a]$

Esercizio 2 - Esempi

L'algoritmo procede confrontando gli estremi dei due vettori. In grassetto sono evidenziati gli elementi messi a confronto ad ogni iterazione:

esempio esito positivo:

$V1 = [a \ c \ d \ e \ b]$ $V2 = [b \ e \ a]$

esempio esito negativo:

$V1 = [a \ c \ d \ e \ b]$ $V2 = [e \ f \ a]$

Esercizio 2 - Esempi

L'algoritmo procede confrontando gli estremi dei due vettori. In grassetto sono evidenziati gli elementi messi a confronto ad ogni iterazione:

esempio esito positivo:

$$V1 = [a c d e b] \quad V2 = [b e a]$$

[**a** c d e b]

[b e **a**] ← i due elementi sono uguali,
accorcio entrambi i vettori

esempio esito negativo:

$$V1 = [a c d e b] \quad V2 = [e f a]$$

[**a** c d e b]

[e f **a**] ← i due elementi sono uguali,
accorcio entrambi i vettori

Esercizio 2 - Esempi

L'algoritmo procede confrontando gli estremi dei due vettori. In grassetto sono evidenziati gli elementi messi a confronto ad ogni iterazione:

esempio esito positivo:

$V1 = [a\ c\ d\ e\ b]$ $V2 = [b\ e\ a]$

[a c d e b]

[b e **a]** ← i due elementi sono uguali,
accorcio entrambi i vettori

[c d e b]

[b **e]** ← i due elementi sono diversi,
accorcio solamente V1

esempio esito negativo:

$V1 = [a\ c\ d\ e\ b]$ $V2 = [e\ f\ a]$

[a c d e b]

[e f **a]** ← i due elementi sono uguali,
accorcio entrambi i vettori

[c d e b]

[e **f]** ← i due elementi sono diversi,
accorcio solamente V1

Esercizio 2 - Esempi

L'algoritmo procede confrontando gli estremi dei due vettori. In grassetto sono evidenziati gli elementi messi a confronto ad ogni iterazione:

esempio esito positivo:

$V1 = [a\ c\ d\ e\ b]$ $V2 = [b\ e\ a]$

[a c d e b] **[b e a]** ← i due elementi sono uguali, accorcio entrambi i vettori

[c d e b] **[b e]** ← i due elementi sono diversi, accorcio solamente V1

[d e b] **[b e]**

esempio esito negativo:

$V1 = [a\ c\ d\ e\ b]$ $V2 = [e\ f\ a]$

[a c d e b] **[e f a]** ← i due elementi sono uguali, accorcio entrambi i vettori

[c d e b] **[e f]** ← i due elementi sono diversi, accorcio solamente V1

[d e b] **[e f]**

Esercizio 2 - Esempi

L'algoritmo procede confrontando gli estremi dei due vettori. In grassetto sono evidenziati gli elementi messi a confronto ad ogni iterazione:

esempio esito positivo:

$V1 = [a\ c\ d\ e\ b]$ $V2 = [b\ e\ a]$

[a c d e b] **[b e a]** ← i due elementi sono uguali, accorcio entrambi i vettori

[c d e b] **[b e]** ← i due elementi sono diversi, accorcio solamente V1

[d e b] **[b e]**

[e b] **[b e]**

esempio esito negativo:

$V1 = [a\ c\ d\ e\ b]$ $V2 = [e\ f\ a]$

[a c d e b] **[e f a]** ← i due elementi sono uguali, accorcio entrambi i vettori

[c d e b] **[e f]** ← i due elementi sono diversi, accorcio solamente V1

[d e b] **[e f]**

[e b] **[e f]**

Esercizio 2 - Esempi

L'algoritmo procede confrontando gli estremi dei due vettori. In grassetto sono evidenziati gli elementi messi a confronto ad ogni iterazione:

esempio esito positivo:

$V1 = [a\ c\ d\ e\ b]$ $V2 = [b\ e\ a]$

[a c d e b] **[b e a]** ← i due elementi sono uguali, accorcio entrambi i vettori

[c d e b] **[b e]** ← i due elementi sono diversi, accorcio solamente V1

[d e b] **[b e]**

[e b] **[b e]**

[b] **[b]**

esempio esito negativo:

$V1 = [a\ c\ d\ e\ b]$ $V2 = [e\ f\ a]$

[a c d e b] **[e f a]** ← i due elementi sono uguali, accorcio entrambi i vettori

[c d e b] **[e f]** ← i due elementi sono diversi, accorcio solamente V1

[d e b] **[e f]**

[e b] **[e f]**

[b] **[e f]**

Esercizio 2 - Esempi

L'algoritmo procede confrontando gli estremi dei due vettori. In grassetto sono evidenziati gli elementi messi a confronto ad ogni iterazione:

esempio *esito positivo*:

$$V1 = [a \ c \ d \ e \ b] \quad V2 = [b \ e \ a]$$

[**a** c d e b] [b e **a**] ← i due elementi sono uguali, accorcio entrambi i vettori

[**c** d e b] [b **e**] ← i due elementi sono diversi, accorcio solamente V1

[**d** e b] [b **e**]

[**e** b] [b **e**]

[**b**] [**b**]

[] [] ← entrambi i vettori hanno lunghezza nulla, vuol dire che V2 è interamente contenuto in V1 in ordine inverso

esempio *esito negativo*:

$$V1 = [a \ c \ d \ e \ b] \quad V2 = [e \ f \ a]$$

[**a** c d e b] [e f **a**] ← i due elementi sono uguali, accorcio entrambi i vettori

[**c** d e b] [e **f**] ← i due elementi sono diversi, accorcio solamente V1

[**d** e b] [e **f**]

[**e** b] [e **f**]

[**b**] [e **f**]

[] [e f] ← V1 è vuoto mentre V2 no. V2 non è contenuto in V1 in ordine inverso.

Esercizio 2 - Caso base e passo induttivo

Caso base:

(con esito positivo): se il vettore v_2 è vuoto, termina con esito positivo 1 (un vettore vuoto, senza elementi, è senz'altro contenuto in qualsiasi altro vettore).

$[]$

$[]$

(con esito negativo): se il vettore v_1 è vuoto ma v_2 non lo è, termina con esito negativo 0 (un vettore non vuoto, che contiene almeno un elemento, non può essere contenuto in un vettore vuoto).

$[]$

$[e f]$

Esercizio 2 - Caso base e passo induttivo

Caso base:

(con esito positivo): se il vettore v_2 è vuoto, termina con esito positivo 1 (un vettore vuoto, senza elementi, è senz'altro contenuto in qualsiasi altro vettore).

$[]$

$[]$

(con esito negativo): se il vettore v_1 è vuoto ma v_2 non lo è, termina con esito negativo 0 (un vettore non vuoto, che contiene almeno un elemento, non può essere contenuto in un vettore vuoto).

$[]$

$[e f]$

Passo induttivo:

se l'elemento iniziale di v_1 è uguale a quello finale di v_2 elimina l'elemento iniziale di v_1 e quello finale di v_2 .

$[a c d e b]$

$[b e a]$

se l'elemento iniziale di v_1 è diverso da quello finale di v_2 elimina soltanto l'elemento iniziale di v_1 .

$[c d e b]$

$[e f]$

Esercizio 2 - Note

Chiamata ricorsiva:

ricorri sui due vettori v_1 e v_2 accorciati in uno dei due modi indicati nel passo induttivo.

Nella formulazione di un programma ricorsivo ci si deve sempre porre il problema della **terminazione**:

nel passo induttivo almeno uno dei due vettori v_1 o v_2 viene sempre accorciato (se non tutti e due) e quindi dopo un numero finito di passi almeno uno dei due vettori sarà vuoto. L'algoritmo termina quando almeno uno dei due vettori è vuoto e questo coincide con uno dei due casi base.

► La terminazione della ricorsione è dunque sempre garantita.

La terminazione DEVE essere garantita altrimenti il programma ciclerebbe all'infinito.

Esercizio 2 - Soluzione

```
function r = contieneInverso(v1,v2)
```

```
end
```

Esercizio 2 - Soluzione

```
function r = contieneInverso(v1,v2)
```

```
if (isempty(v2))
```

```
    r = 1;
```

▶ caso base positivo

```
end
```

Esercizio 2 - Soluzione

```
function r = contieneInverso(v1,v2)

if (isempty(v2))
    r = 1;           ▶ caso base positivo
else if (isempty(v1))
    r = 0;           ▶ caso base negativo

end
```

Esercizio 2 - Soluzione

```
function r = contieneInverso(v1,v2)

if (isempty(v2))
    r = 1;           ▶ caso base positivo
else if (isempty(v1))
    r = 0;           ▶ caso base negativo
else
    if (v1(1) == v2(end))
        r = contieneInverso(v1(2:end),v2(1:end-1)); ▶ passo induttivo
    else
        ▶ positivo
    end
end
```

Esercizio 2 - Soluzione

```
function r = contieneInverso(v1,v2)

if (isempty(v2))
    r = 1;           ▶ caso base positivo
else if (isempty(v1))
    r = 0;           ▶ caso base negativo
else
    if (v1(1) == v2(end))
        r = contieneInverso(v1(2:end),v2(1:end-1)); ▶ passo induttivo
    else
        r = contieneInverso(v1(2:end),v2(1:end));   ▶ passo induttivo
    end
end
end
end
```

Esercizio 2 - Esempio di invocazione

```
v1 = input('inserire il vettore V1: ');
v2 = input('inserire il vettore V2: ');

r = contieneInverso(v1,v2);

if (r)
    disp(['tutti gli elementi di V2 sono contenuti in ordine inverso
in V1']);
else
    disp(['gli elementi di V2 non sono contenuti in ordine inverso in
V1']);
end
```

Esercizio 3 - Da tenere a mente

Esercizio 3 - Da tenere a mente

bit_indirizzo_fisico = **NPF** + *bit_offset*

Esercizio 3 - Da tenere a mente

bit_indirizzo_fisico = **NPF** + *bit_offset*

bit_indirizzo_virtuale = **NPV** + *bit_offset*

Esercizio 3 - Da tenere a mente

bit_indirizzo_fisico = **NPF** + *bit_offset*

bit_indirizzo_virtuale = **NPV** + *bit_offset*

***Ricorda:** potrebbe essere la stessa pagina se e solo se i **bit_offset** sono uguali, altrimenti è sicuramente diversa!*

Esercizio 3 - Da tenere a mente

bit_indirizzo_fisico = **NPF** + *bit_offset*

bit_indirizzo_virtuale = **NPV** + *bit_offset*

***Ricorda:** potrebbe essere la stessa pagina se e solo se i **bit_offset** sono uguali, altrimenti è sicuramente diversa!*

Esercizio 3 - Da tenere a mente

bit_indirizzo_fisico = **NPF** + **bit_offset**

bit_indirizzo_virtuale = **NPV** + **bit_offset**

***Ricorda:** potrebbe essere la stessa pagina se e solo se i **bit_offset** sono uguali, altrimenti è sicuramente diversa!*

dimensione_pagina = $2^{\text{bit_offset}}$

Esercizio 3 - Da tenere a mente

bit_indirizzo_fisico = **NPF** + **bit_offset**

bit_indirizzo_virtuale = **NPV** + **bit_offset**

***Ricorda:** potrebbe essere la stessa pagina se e solo se i **bit_offset** sono uguali, altrimenti è sicuramente diversa!*

dimensione_pagina = $2^{\text{bit_offset}}$

pagine_fisiche_indirizzabili = 2^{NPF}

Esercizio 3 - Da tenere a mente

bit_indirizzo_fisico = **NPF** + **bit_offset**

bit_indirizzo_virtuale = **NPV** + **bit_offset**

***Ricorda:** potrebbe essere la stessa pagina se e solo se i **bit_offset** sono uguali, altrimenti è sicuramente diversa!*

dimensione_pagina = $2^{\text{bit_offset}}$

pagine_fisiche_indirizzabili = 2^{NPF}

pagine_virtuali_indirizzabili = 2^{NPV}

Esercizio 3 - Da tenere a mente

bit_indirizzo_fisico = **NPF** + **bit_offset**

bit_indirizzo_virtuale = **NPV** + **bit_offset**

***Ricorda:** potrebbe essere la stessa pagina se e solo se i **bit_offset** sono uguali, altrimenti è sicuramente diversa!*

dimensione_pagina = $2^{\text{bit_offset}}$

pagine_fisiche_indirizzabili = 2^{NPF}

pagine_virtuali_indirizzabili = 2^{NPV}

Esercizio 3 - Da tenere a mente

$$\text{bit_indirizzo_fisico} = \text{NPF} + \text{bit_offset}$$

$$\text{bit_indirizzo_virtuale} = \text{NPV} + \text{bit_offset}$$

***Ricorda:** potrebbe essere la stessa pagina se e solo se i **bit_offset** sono uguali, altrimenti è sicuramente diversa!*

$$\text{dimensione_pagina} = 2^{\text{bit_offset}}$$

$$\text{pagine_fisiche_indirizzabili} = 2^{\text{NPF}}$$

$$\text{pagine_virtuali_indirizzabili} = 2^{\text{NPV}}$$

$$\text{dimensione_memoria_fisica} = 2^{\text{bit_indirizzo_fisico}}$$

Esercizio 3 - Da tenere a mente

$$\text{bit_indirizzo_fisico} = \text{NPF} + \text{bit_offset}$$

$$\text{bit_indirizzo_virtuale} = \text{NPV} + \text{bit_offset}$$

***Ricorda:** potrebbe essere la stessa pagina se e solo se i **bit_offset** sono uguali, altrimenti è sicuramente diversa!*

$$\text{dimensione_pagina} = 2^{\text{bit_offset}}$$

$$\text{pagine_fisiche_indirizzabili} = 2^{\text{NPF}}$$

$$\text{pagine_virtuali_indirizzabili} = 2^{\text{NPV}}$$

$$\text{dimensione_memoria_fisica} = 2^{\text{bit_indirizzo_fisico}}$$

$$\text{dimensione_memoria_virtuale} = 2^{\text{bit_indirizzo_virtuale}}$$

Esercizio 3 - Soluzioni

Esercizio 3 - Soluzioni

c) **12 bit** indirizzo **fisico**, **16 pagine fisiche**

Esercizio 3 - Soluzioni

- c) **12 bit** indirizzo **fisico**, **16 pagine fisiche**
NPF: 4 bit, offset: 8 bit

Esercizio 3 - Soluzioni

- c) **12 bit** indirizzo **fisico**, **16 pagine fisiche**
NPF: 4 bit, **offset: 8 bit**
64 pagine virtuali, **stesso offset**

Esercizio 3 - Soluzioni

- c) **12 bit** indirizzo **fisico**, **16 pagine fisiche**
NPF: 4 bit, **offset: 8 bit**
64 pagine virtuali, **stesso offset**
14 bit indirizzo virtuale

Esercizio 3 - Soluzioni

- c) **12 bit** indirizzo **fisico**, **16 pagine fisiche**
NPF: 4 bit, **offset: 8 bit**
64 pagine virtuali, **stesso offset**
14 bit indirizzo virtuale
NPV: 6 bit, **offset: 8 bit**

Esercizio 3 - Soluzioni

- c) **12 bit** indirizzo **fisico**, **16 pagine fisiche**
NPF: 4 bit, **offset: 8 bit**
64 pagine virtuali, **stesso offset**
14 bit indirizzo virtuale
NPV: 6 bit, **offset: 8 bit**

Esercizio 3 - Soluzioni

c) **12 bit** indirizzo **fisico**, **16 pagine fisiche**

NPF: 4 bit, **offset: 8 bit**

64 pagine virtuali, **stesso offset**

14 bit indirizzo virtuale

NPV: 6 bit, **offset: 8 bit**

a) **pagina fisica** da **256 byte**

Esercizio 3 - Soluzioni

c) **12 bit** indirizzo **fisico**, **16 pagine fisiche**

NPF: 4 bit, **offset: 8 bit**

64 pagine virtuali, **stesso offset**

14 bit indirizzo virtuale

NPV: 6 bit, **offset: 8 bit**

a) **pagina fisica** da **256 byte**

memoria fisica da **4Kbyte**

Esercizio 3 - Soluzioni

c) **12 bit** indirizzo **fisico**, **16 pagine fisiche**

NPF: 4 bit, **offset: 8 bit**

64 pagine virtuali, **stesso offset**

14 bit indirizzo virtuale

NPV: 6 bit, **offset: 8 bit**

a) **pagina fisica** da **256 byte**

memoria fisica da **4Kbyte**

Esercizio 3 - Soluzioni

c) **12 bit** indirizzo **fisico**, **16 pagine fisiche**

NPF: 4 bit, **offset: 8 bit**

64 pagine virtuali, **stesso offset**

14 bit indirizzo virtuale

NPV: 6 bit, **offset: 8 bit**

a) **pagina fisica** da **256 byte**

memoria fisica da **4Kbyte**

b) **pagina virtuale** da **256 byte**

Esercizio 3 - Soluzioni

c) **12 bit** indirizzo **fisico**, **16 pagine fisiche**

NPF: 4 bit, **offset: 8 bit**

64 pagine virtuali, **stesso offset**

14 bit indirizzo virtuale

NPV: 6 bit, **offset: 8 bit**

a) **pagina fisica** da **256 byte**

memoria fisica da **4Kbyte**

b) **pagina virtuale** da **256 byte**

memoria virtuale da **16Kbyte**