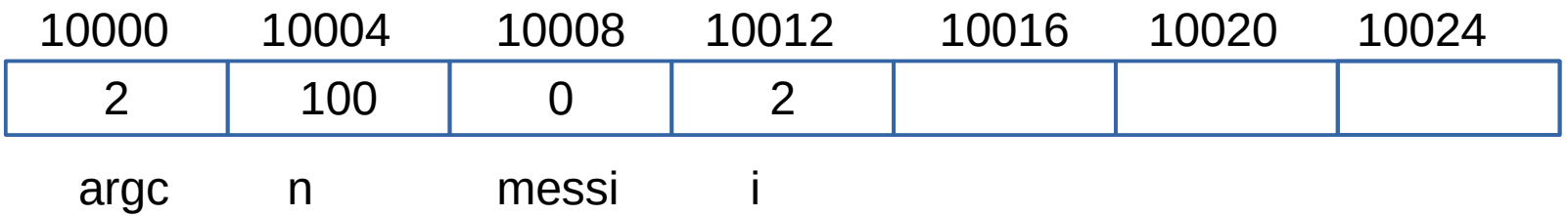
Lezione 3

Introduzione ai puntatori

Uno sguardo alla RAM

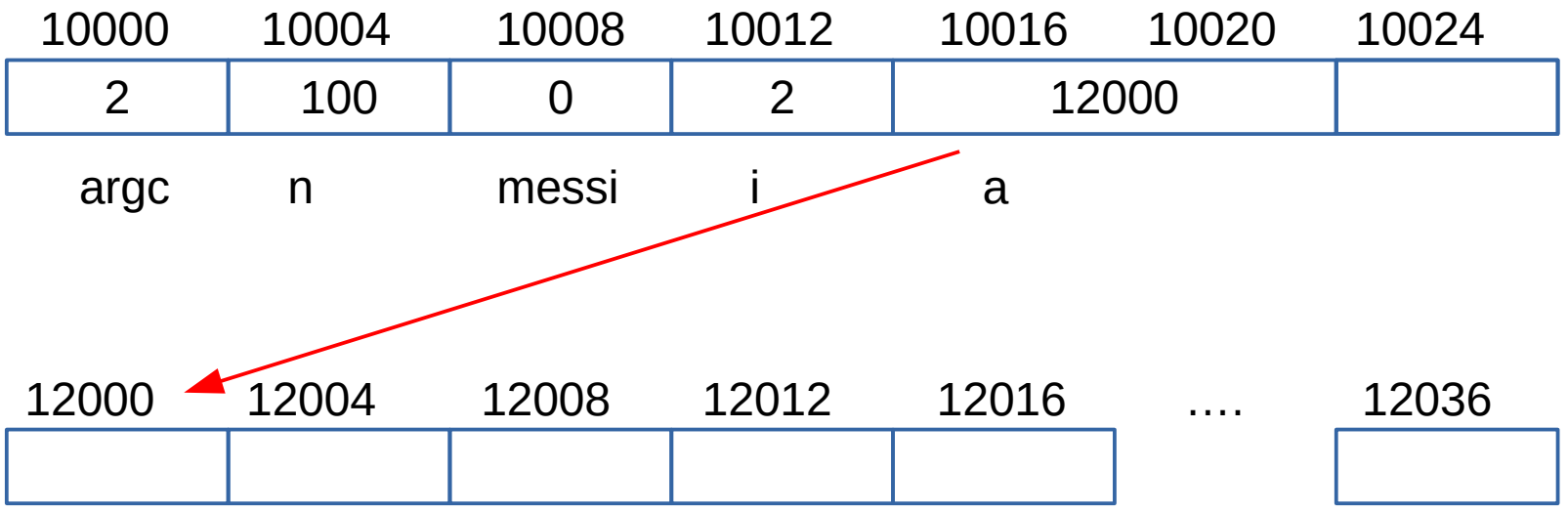
- La RAM del computer consiste in bytes, ad esempio 512MB = 536870912, singolarmente indirizzabili
- Ogni byte è individuato da un indirizzo, tra 0 e 536870911
- Il nostro programma occupa una porzione di questa RAM, diciamo a partire dalla posizione 10,000

Supponiamo il nostro programma contenga delle variabili intere (4 byte)



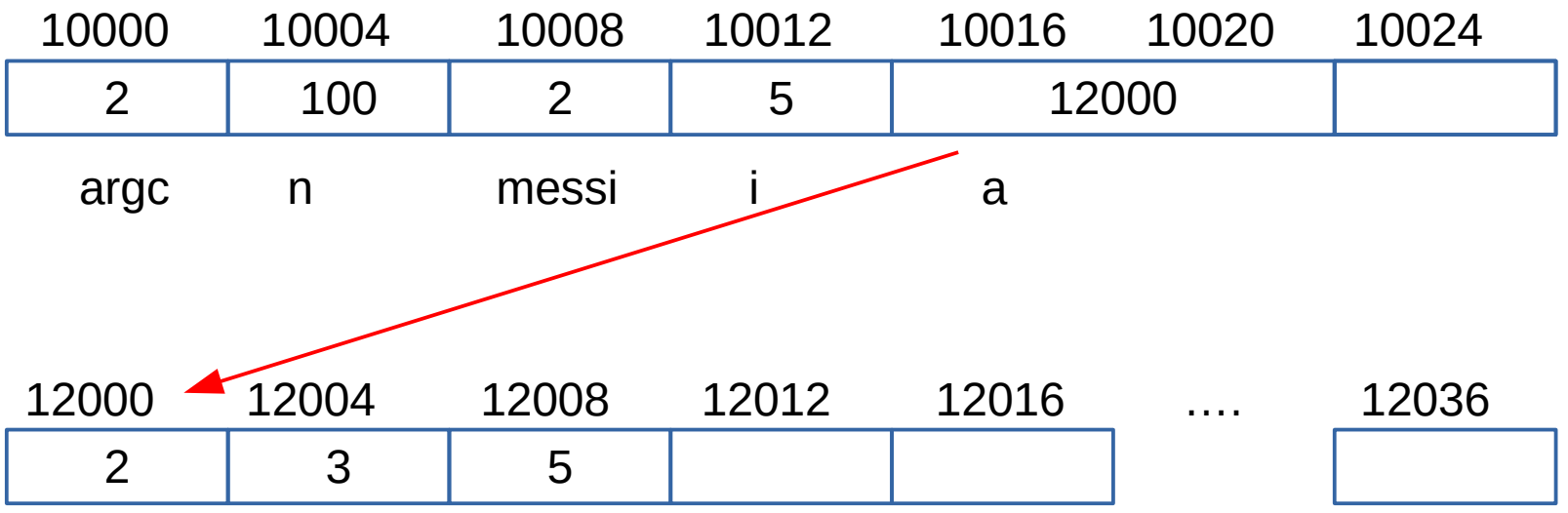
Situazione prima della chiamata a **malloc()**: solo variabili intere

Supponiamo il nostro programma contenga delle variabili intere (4 byte)



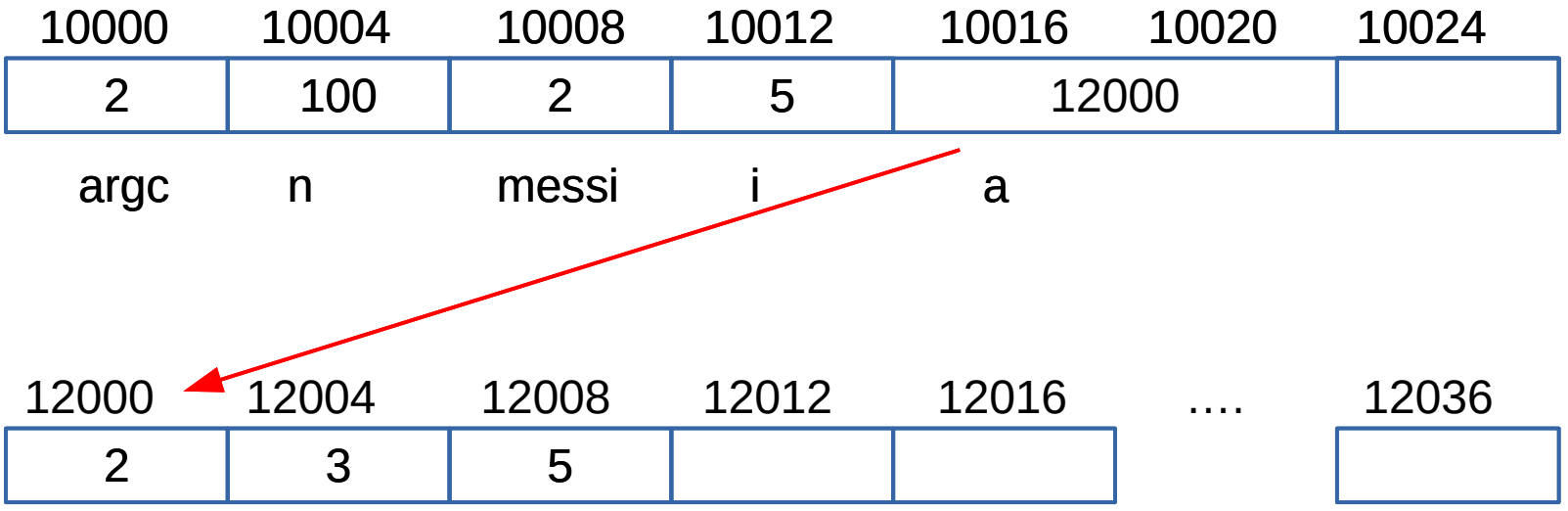
Dopo la **malloc**: 40 byte riservati dal sistema per il mio array

Supponiamo il nostro programma contenga delle variabili intere (4 byte)

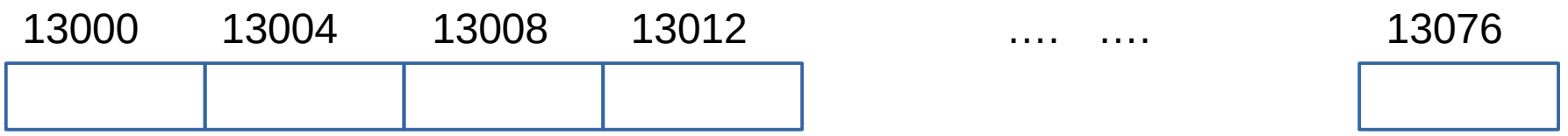


Dopo aver scritto i primi tre elementi nell’array a[ ]

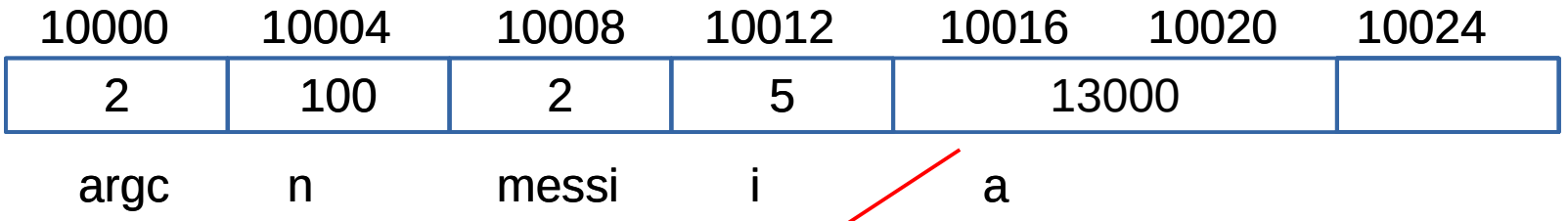
realloc e free



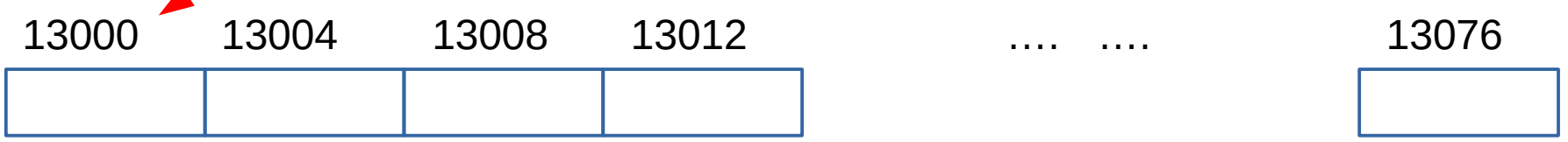
esecuzione di **a = realloc(a,80)**



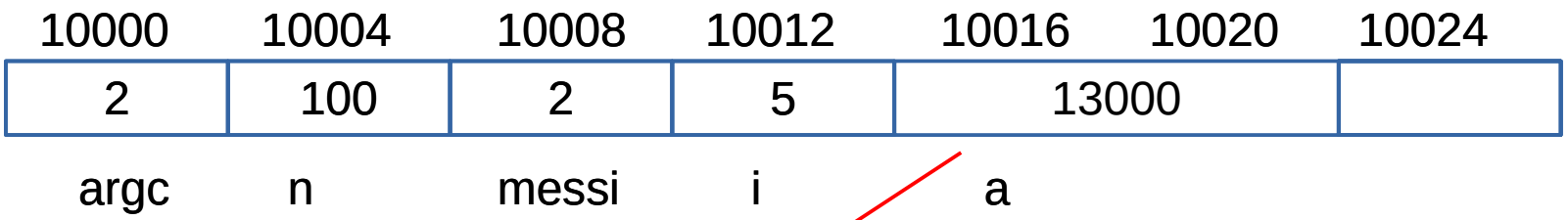
realloc e free



dopo esecuzione di **a = realloc(a,80)**



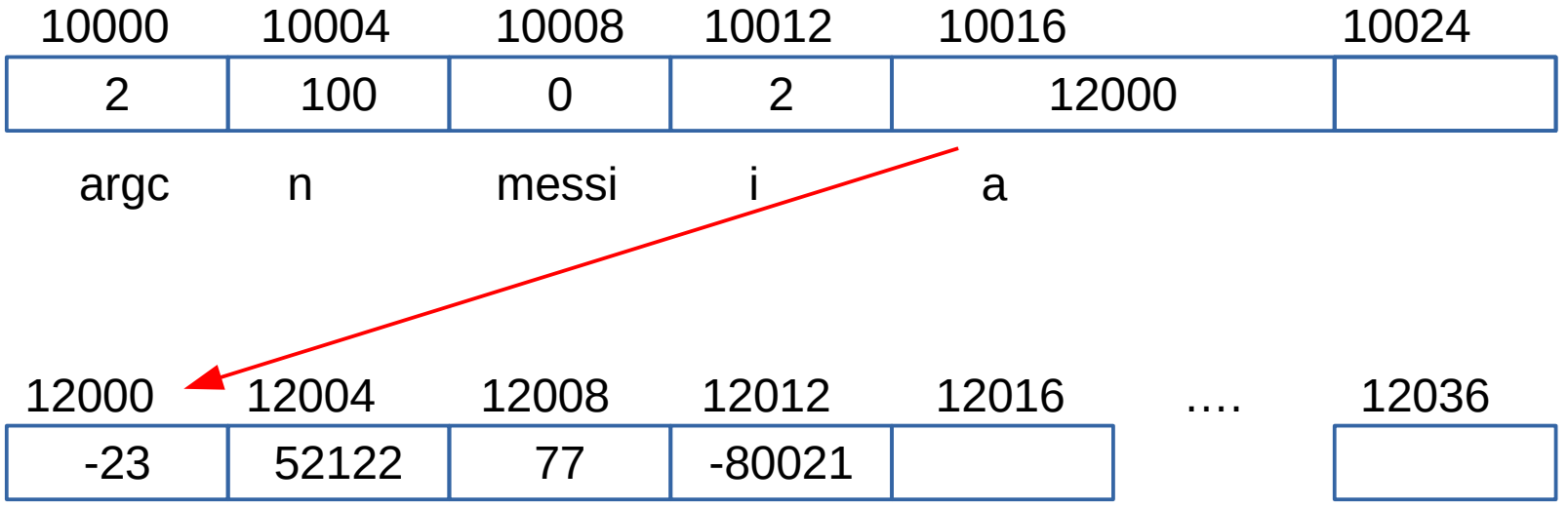
realloc e free



dopo esecuzione di **free(a)**

?

La memoria restituita da malloc non è inizializzata!



Dopo la malloc: 40 byte riservati dal sistema per il mio array

Da ricordare

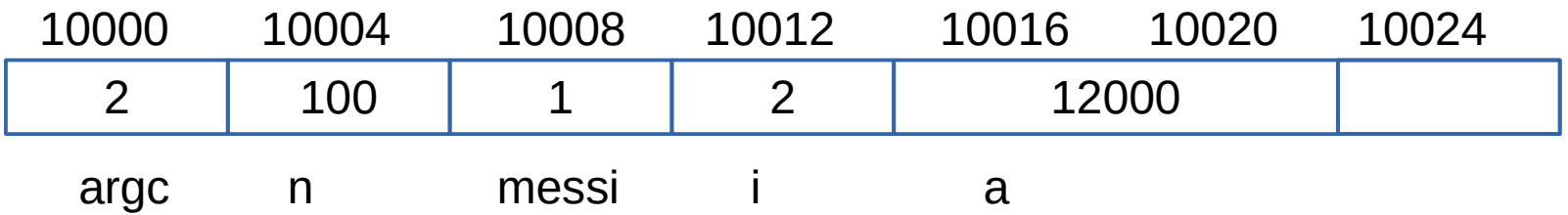
- **malloc** serve per creare un array
- **realloc** per cambiarne le dimensioni
- **free** per distruggerlo (deallocarlo)
- un array è individuato dalla posizione in memoria del suo primo elemento
- tocca a noi non accedere **mai** a zone di memoria non assegnate a noi

Puntatori

- una variabile che contiene l’indirizzo in memoria di un’altra variabile è detta ***puntatore***
- il tipo di un puntatore è individuato dal tipo della variabile puntata seguito ad un **`*`**. Ad esempio: **`int *`**
- I puntatori contengono tutti la stessa cosa (un indirizzo) ma il compilatore distingue tra **`int *`**, **`long *`**, **`double *`**, etc.

Estrarre l’indirizzo

- un altro modo per generare puntatori è l’operatore **`&`**
- data una variabile **`w`** se scrivo **`&w`** ottengo l’indirizzo di **`w`** nella RAM



Esempi: &messi → 10008, &a → 10016

Cosa serve l’indirizzo?

- si usa solo con l’operatore ***** (dereferenziazione)
- esempio molto importante:

```
// dichiaro p puntatore a intero senza inizializzarlo
int *p;
// metto in p l’indirizzo in RAM di messi
p = &messi;
// incremento n del valore della variabile a cui punta p (= messi)
n += *p;
// scrivo 7 nella variabile a cui punta p (= messi)
*p = 7;   // l’effetto è lo stesso di scrivere messi=7
p = &n;  // ora invece cambio il valore di p
```

Cosa serve l’indirizzo?

Gli operatori **&** e * servono per “aggirare” il fatto che nel C quando chiamo una funzione i parametri sono passati per valore

```
main() {
  …
  int n=7;
  fun(n);
  …
}
```

```
int fun(int a) {
  …
  x = a*a;
  a += x*y+a;
  …
}
```

nulla di ciò che viene fatto in **fun** può alterare il valore di **n**

Cosa serve l’indirizzo?

Gli operatori **&** e * servono per “aggirare” il fatto che nel C quando chiamo una funzione i parametri sono passati per valore

```
main() {
  …
  int n=7;
  zun(&n);
  …
}
```

```
int zun(int *p) {
  …
  *p=5;
  …
}
```

l’esecuzione di *p=5 scrive il valore 5 dentro **n**