



UNIVERSITÀ  
DEGLI STUDI  
DI PADOVA



DIPARTIMENTO  
**MATEMATICA**

# CPUSim

## Laboratorio di Architettura degli Elaboratori

Corso di Laurea in Informatica A.A. 2019/2020

---

Nicolò Navarin

Davide Rigoni

[nnavarin@math.unipd.it](mailto:nnavarin@math.unipd.it)

15 Dicembre 2020

- Scaricare il simulatore dal moodle del corso (CPUSim4.0.11.zip);
- Estrarre l'archivio (ricordatevi dove);
- Avviare il simulatore:
  - doppio click;
  - da console, posizionandosi nella cartella del simulatore, eseguire il comando (**Nota:** tutto sulla stessa riga):
    - macOS/Linux

```
java -cp .:richtextfx-0.6.10.jar:reactfx  
-2.0-M4.jar -jar CPUSim-4.0.11.jar
```

- Windows

```
java -cp .;richtextfx-0.6.10.jar;reactfx  
-2.0-M4.jar -jar CPUSim-4.0.11.jar
```

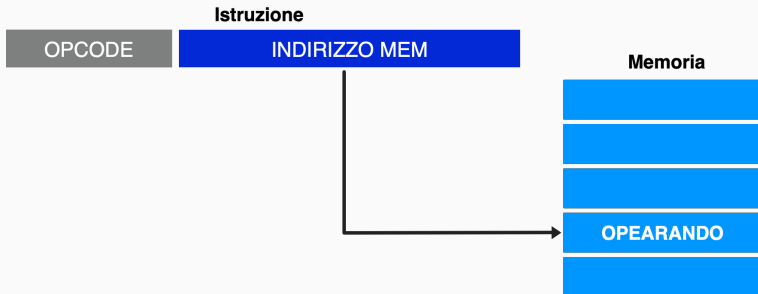
# Metodi di indirizzamento

- **Immediato**: valore operando nell'istruzione (no indirizzi);
- **Diretto**: campo indirizzo = indirizzo dell'operando;
- **Indiretto**: campo indirizzo = indirizzo di una cella di M che contiene l'indirizzo dell'operando;
- **Registro**: l'operando è in un registro specificato nell'istruzione;
- **Registro indiretto**: il registro specificato nell'istruzione contiene l'indirizzo di M dell'operando;
- **Spiazzamento**: due campi (A,R) - **A**: indirizzo di base (diretto) + **R**: registro che contiene un valore da sommare ad A per ottenere l'indirizzo dell'operando;
- **Pila** (sequenza lineare di locazioni riservate di M): il registro *Stack Pointer* contiene l'indirizzo della cima della pila (in cui si trova l'operando).

# Indirizzamento diretto - Wombat1

Campo indirizzo  $\Rightarrow$  indirizzo dell'operando nella memoria:

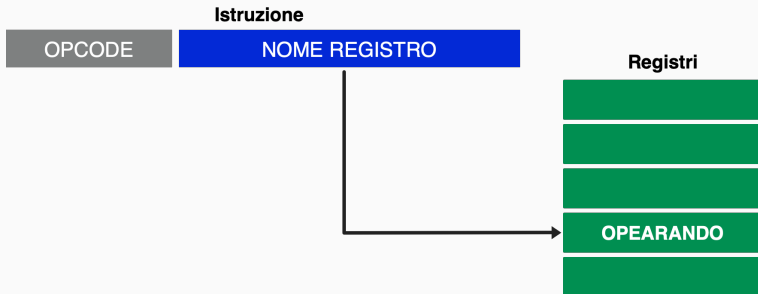
- CPU Wombat1;
- e.g. `ADD X ( $acc + Mem[X] \rightarrow acc$ );`
- **1 accesso** alla memoria all'indirizzo X.



# Indirizzamento a registro - Wombat2

Campo registro  $\Rightarrow$  registro in cui si trova l'operando:

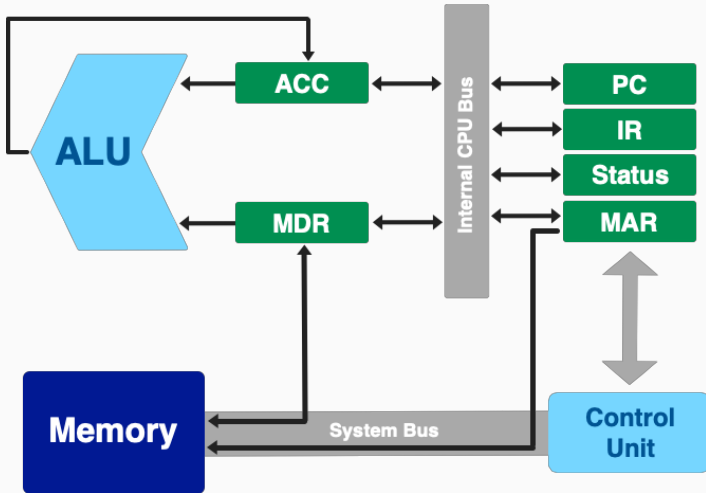
- CPU Wombat2;
- e.g. `writeR X (R[X]  $\rightarrow$  output)`;
- **nessun accesso** alla memoria.



Oggi definiremo una nuova CPU che utilizza l'indirizzamento indiretto:

- Apriamo Wombat1.cpu;
- Salviamo con un nuovo nome, e.g., **Wombat3.cpu**;
- Se abbiamo altre istruzioni oltre quelle di default, le cancelliamo (opcode da C in su).

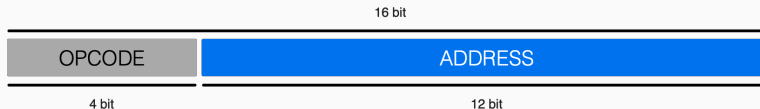
## Wombat3 - architettura (come Wombat1!)



## Wombat3 - definizione di nuove (micro)istruzioni

Definiremo nuove istruzioni che interpretano gli operandi non come indirizzi di memoria da caricare, ma come **indirizzi di indirizzi di memoria** da caricare.

### Formato istruzioni



Servirà una nuova microistruzione:

```
mdr(4-15) -> mar
```

che copia i 12 bit meno significativi di `mdr` in `mar`.

Poi, definiamo `load`, `store` e `add` con indirizzamento indiretto.

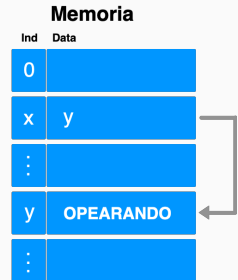
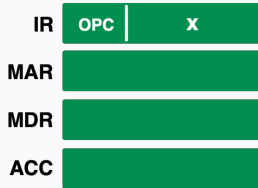


## Nuova istruzione - loadInd

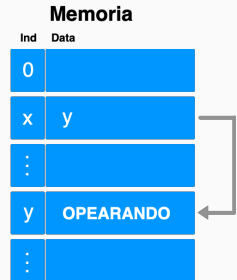
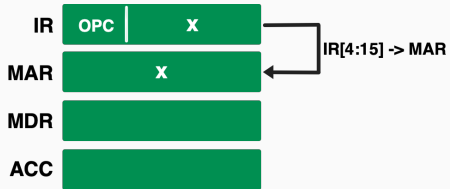
```
loadInd addr
```

- *load indiretta*: `addr` è l'indirizzo dell'indirizzo della locazione di memoria del valore da caricare nell'accumulatore;
- *Modify* → *Machine Instructions*;
- duplichiamo la *load* e la modifichiamo.

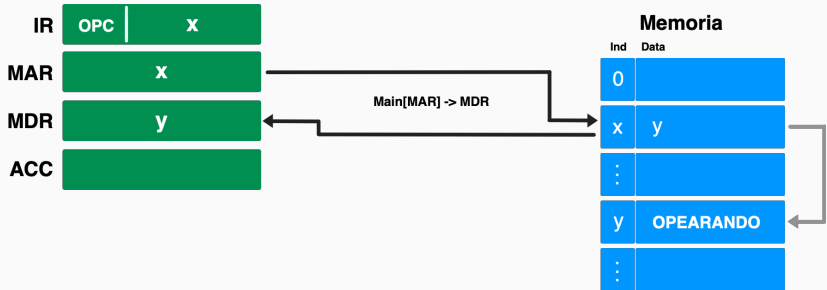
# loadInd - step by step (0/5)



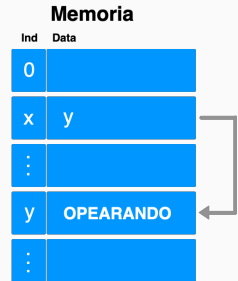
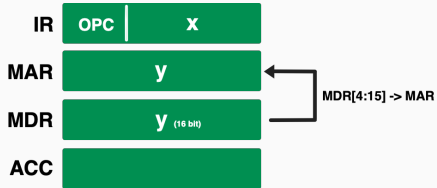
# loadInd - step by step (1/5)



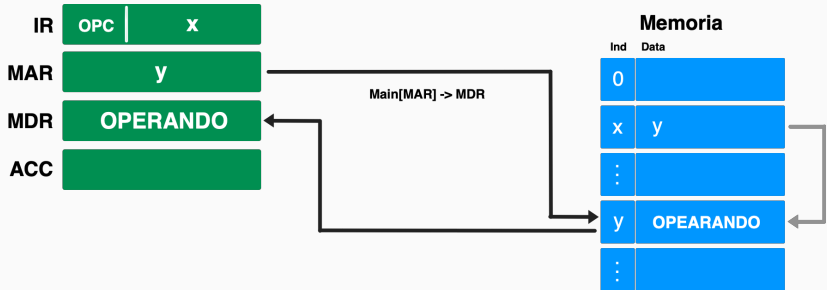
## loadInd - step by step (2/5)



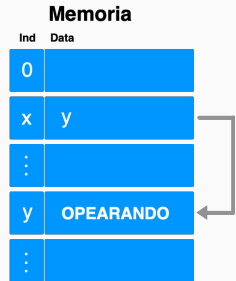
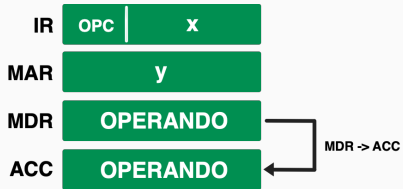
## loadInd - step by step (3/5)



## loadInd - step by step (4/5)



## loadInd - step by step (5/5)



## Microistruzioni:

```
ir(4-15) -> mar  
Main[mar] -> mdr  
mdr(4-15) -> mar  
Main[mar] -> mdr  
mdr -> acc  
End
```



## Nuova istruzione - storeInd

```
storeInd addr
```

- *store indiretta*: `addr` è l'indirizzo dell'indirizzo della locazione di memoria dove scrivere il contenuto dell'accumulatore.

## Nuova istruzione - storeInd

**storeInd** addr

- *store indiretta*: addr è l'indirizzo dell'indirizzo della locazione di memoria dove scrivere il contenuto dell'accumulatore.

### Microistruzioni:

```
ir(4-15) -> mar  
Main[mar] -> mdr  
mdr(4-15) -> mar  
acc -> mdr  
mdr -> Main[mar]  
End
```

## Nuova istruzione - addInd

**addInd** addr

- *add indiretta*: addr è l'indirizzo dell'indirizzo della locazione di memoria del valore da sommare all' accumulatore.

## Nuova istruzione - addInd

**addInd** addr

- *add indiretta*: addr è l'indirizzo dell'indirizzo della locazione di memoria del valore da sommare all' accumulatore.

### Microistruzioni:

```
ir(4-15) -> mar  
Main[mar] -> mdr  
mdr(4-15) -> mar  
Main[mar] -> mdr  
acc + mdr -> acc  
End
```

# Wombat3 - instruction set

- **READ**: legge un intero da input e lo mette in ACC
- **WRITE**: scrive in output il contenuto di ACC
- **LOAD X**: dalla cella di memoria X al registro ACC
- **STORE X**: da registro ACC alla cella di memoria X
- **ADD X**: somma ACC e il contenuto della cella di mem. X e mette in ACC
- **SUBTRACT X, MULTIPLY X, DIVIDE X** (divisione intera)
- **JUMP X**: salta all'istruzione con etichetta X
- **JMPZ X**: salta all'istruzione X se  $ACC = 0$
- **JMPN X**: salta all'istruzione X se  $ACC < 0$
- **STOP**: segnala la fine del programma
- **LOADIND X**: dalla cella di memoria indirizzata dal valore contenuto nella cella di memoria X ad ACC
- **STOREIND X**: da ACC alla cella di memoria indirizzata dal valore contenuto nella cella di memoria X
- **ADDIND X**: somma il contenuto di ACC e della cella di memoria indirizzata dal valore della cella di memoria X, e mette il risultato in ACC.

## Esercizio 1

Utilizzando istruzioni ad indirizzamento indiretto, scrivere un programma che legge una sequenza di interi e li somma finché non legge un numero negativo. Alla fine stampa su output la somma (senza includere l'ultimo numero negativo).

**Nota:** *se definiamo la locazione di memoria per la somma ad un indirizzo occupato dal codice del programma, succedono cose strane (il codice viene sovrascritto)!*

## Esercizio 2

Sfruttando le istruzioni ad indirizzamento indiretto, scrivere un programma che legge un indirizzo da input e azzerava il contenuto della locazione di memoria corrispondente.

**Nota:** *senza le istruzioni ad indirizzamento indiretto non era possibile!*

## Esercizio 3

Scrivere un programma che legge due interi, chiamiamoli `ind` e `cont`. Poi, sfruttando le istruzioni ad indirizzamento indiretto, scrive nell'indirizzo di memoria `ind` e nei `cont` successivi i valori `cont, cont - 1, ..., 0`, rispettivamente.

*Esempio:* al termine dell'esecuzione del programma con `ind = 20` e `cont = 3`, la situazione della memoria sarà la seguente:

ind	data
0	...
...	...
20	3
22	2
24	1
26	0
...	...