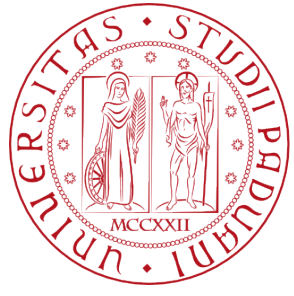
CPUSim - 3

Laboratorio 21/12/2016

Tommaso Padoan

e-mail: padoan@math.unipd.it



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

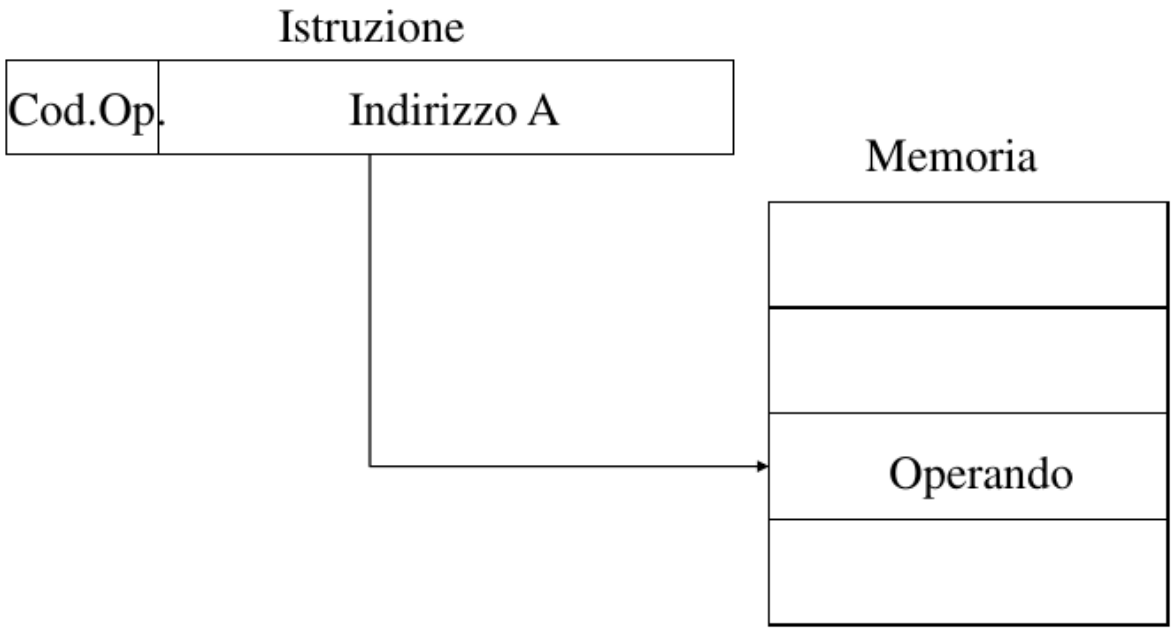
Modi di indirizzamento

- 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: 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, registro *Stack Pointer* contiene l'indirizzo della cima della pila, operando sempre in cima)

Modi di indirizzamento visti in lab - 1

Diretto (campo indirizzo = indirizzo dell'operando nella memoria)

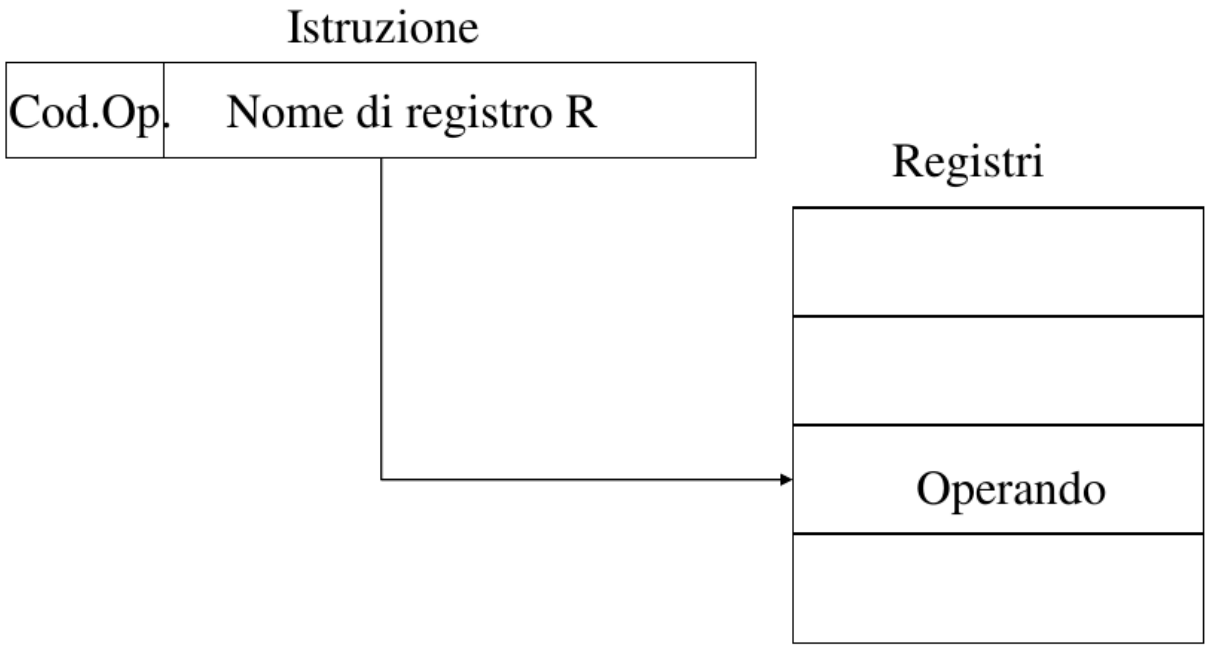
- CPU Wombat1
- e.g. ADD X ($acc + Mem[X] \rightarrow acc$)
- 1 accesso alla memoria, all'indirizzo X.



Modi di indirizzamento visti in lab - 2

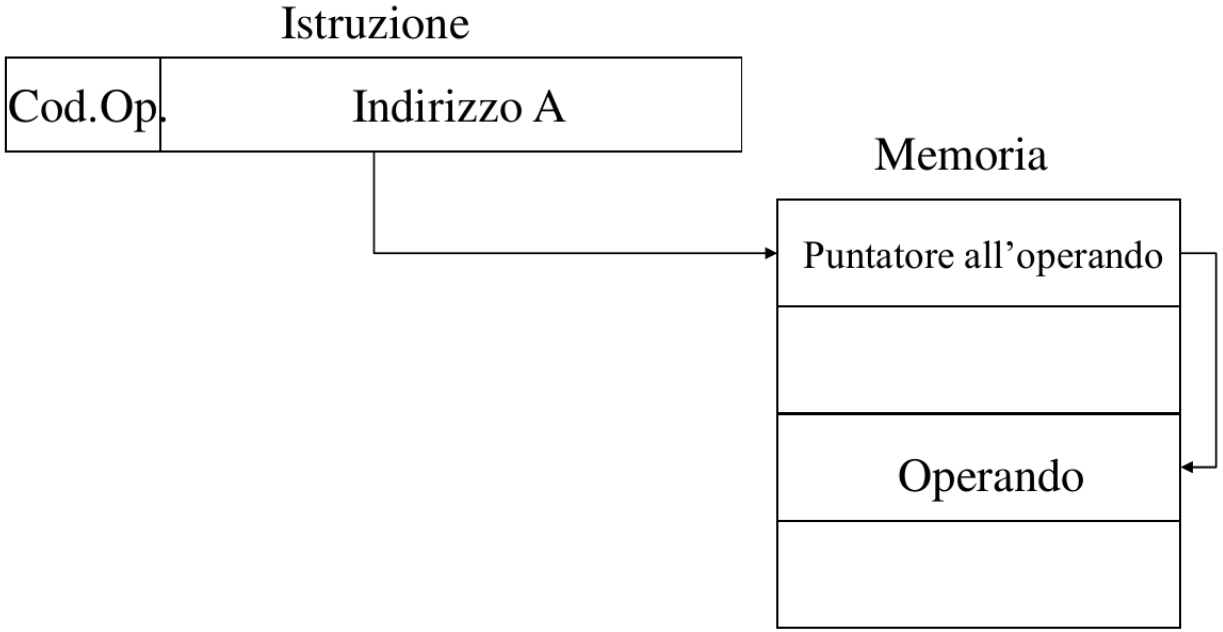
Registro (campo registro = registro in cui si trova l'operando)

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

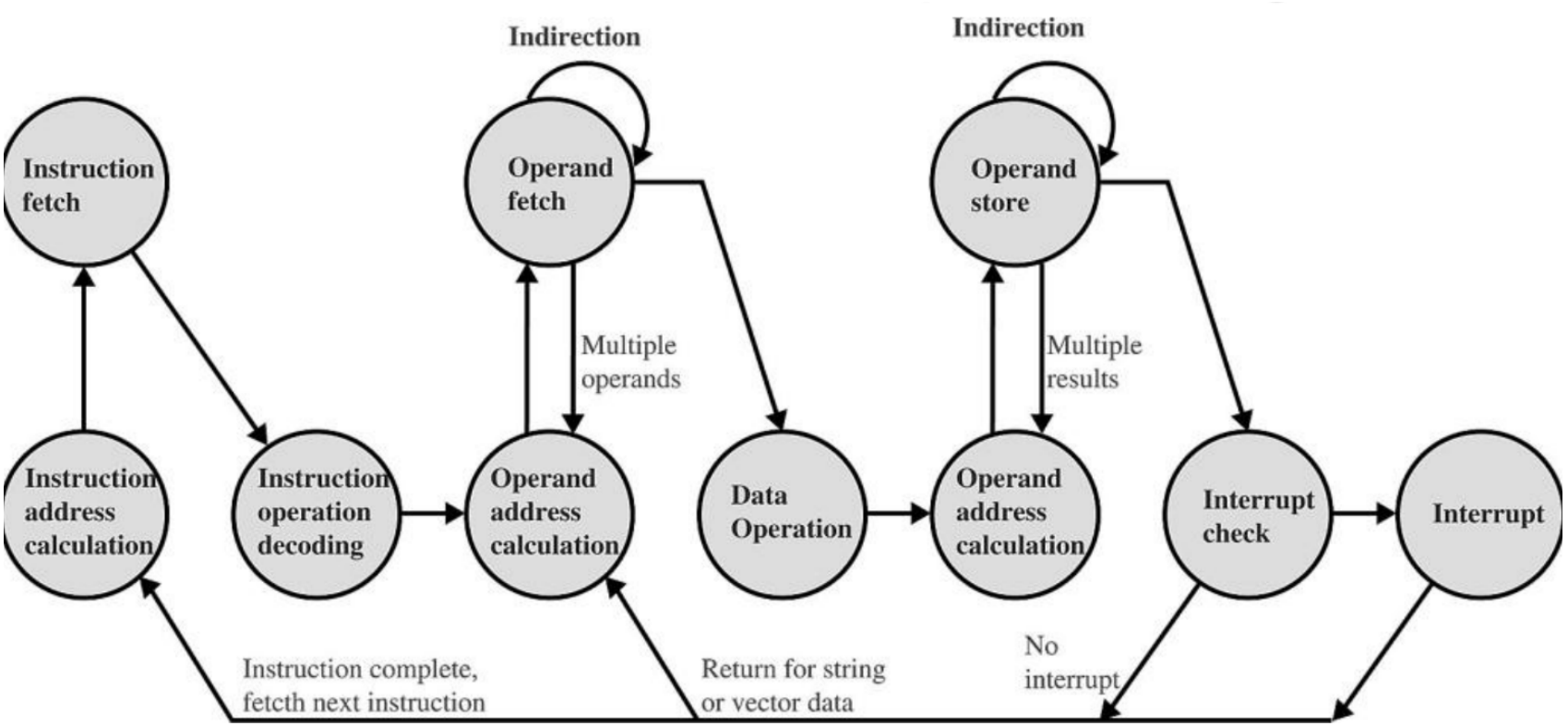


Oggi vedremo...

Indiretto (campo indirizzo = indirizzo di una cella di memoria, che contiene l'indirizzo della cella di memoria contenente l'operando).

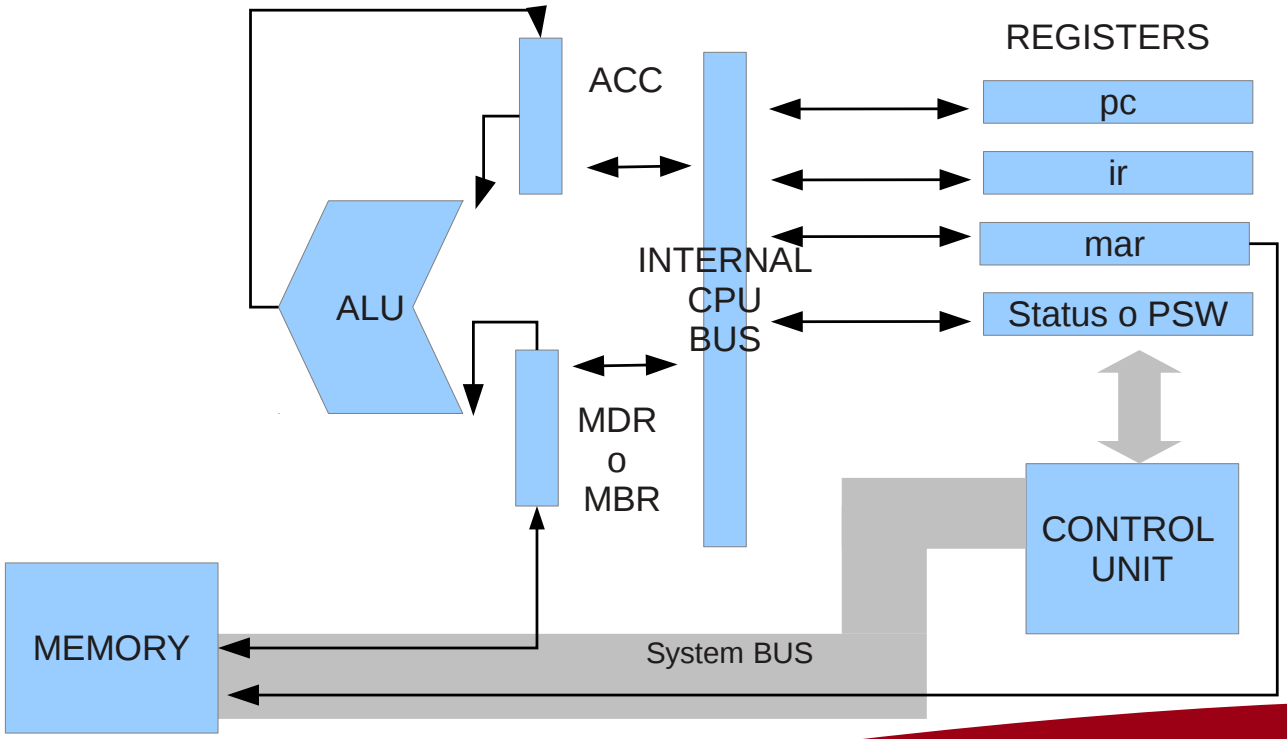


Ind. indiretto - Ciclo di esecuzione



Indirizzamento indiretto: Wombat3

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



Indirizzamento indiretto: Wombat3

Definire nuove istruzioni che interpretino gli operandi:

- *non* come indirizzi di memoria da caricare, ma
- come indirizzi degli indirizzi di memoria da caricare.

Sintassi: **op** (4 bit) + **address** (del puntatore, 12 bit).

Servirà una nuova microistruzione:

- $mdr(4-15)->mar$, copia i 12 bit meno significativi di mdr in mar.

Definiamo *Load*, *Store* e *Add* con indirizzamento indiretto.

Nuove Istruzioni - loadInd

loadInd

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

loadInd

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

Nuove Istruzioni - storeInd

storeInd

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

Nuove Istruzioni - storeInd

storeInd

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

storeInd

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

Nuove Istruzioni - addInd

addInd

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

Nuove Istruzioni - addInd

addInd

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

addInd

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

Salviamo la CPU Wombat3.

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 il contenuto di ACC e della cella di memoria X, e mette il risultato 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, il programma 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

Utilizzando istruzioni ad indirizzamento indiretto, il programma legge due interi, chiamiamoli *ind* e *cont*. Poi, scrive nell'indirizzo di memoria *ind* e nei *cont* successivi i valori $cont, cont - 1, \ldots, 0$ rispettivamente.

E.g. al termine dell'esecuzione del programma con $ind = 20$ e $cont = 3$, la situazione della memoria sarà la seguente:

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