UNIVERSITÀ DEGLI STUDI DI MILANO
matteo.re@unimi.it
https://homes.di.unimi.it/re/arch2-lab-2015-2016.html

DIPARTIMENTO DI INFORMATICA
Laboratorio di Architetture degli Elaboratori II
Turno B : (G – Z)

9b

Strutture dati II

SOMMARIO:

1) Allocazione dinamica memoria (ripasso)
2) Heap (ripasso)
3) Chiamata di sistema sbrk (ripasso)
4) Liste concatenate
5) Liste concatenate, costruzione in memoria statica
6) Liste concatenate, allocazione dinamica memoria per i nodi
7) Nodo corrente ed estensione della lista concatenata
8) Costruzione lista concatenata mediante cicli
9) Attraversamento lista concatenata e stampa del suo contenuto
10) Esercizi

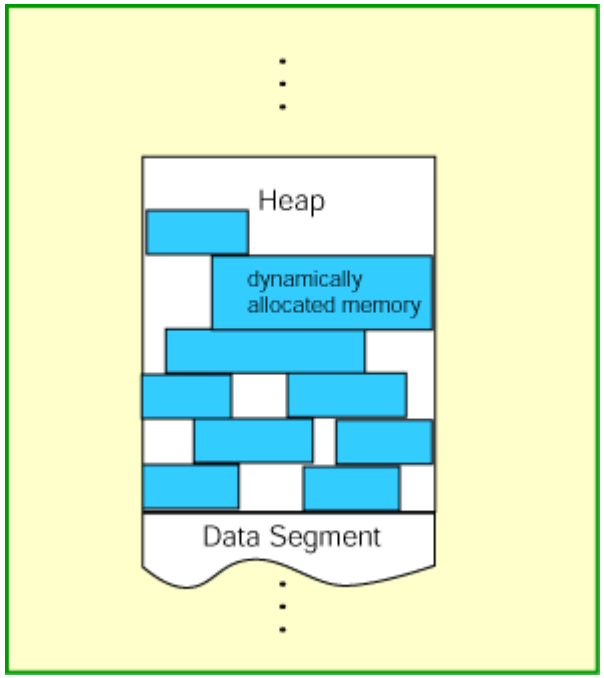
Ultimo aggiornamento: 12/5/2016

L’allocazione dinamica della memoria si verifica quando un programma richiede al sistema operativo un blocco di memoria dichiarando, preventivamente, la dimensione dello stesso.

Il blocco ricevuto può essere utilizzato in vari modi:

. Storage di un singolo valore
. Storage di diversi tipi di dati (ad es. interi, float …) all’interno di un’unica struttura dati

La memoria può essere restituita al sistema quando non è più necessaria al programma e non ci sono vincoli riguardo all’ordine in cui la memoria allocata viene restituita al sistema (differenza rispetto a quanto avviene quando si utilizza lo stack).



I blocchi di memoria richiesti dal programma vengono dallo heap, ossia la regione posta al di sopra della regione contenente i dati statici del programma.

A differenza dello stack che viene gestito tramite una logica di tipo LIFO lo heap (letteralmente «mucchio») assomiglia più ad una libreria in cui i libri vengono prelevati e ricollocati creando un pattern di arree utilizzate e non utilizzate.

Il modo in cui si effettua la richiesta per l'allocazione dinamica di un blocco di memoria è il seguente:

```
li      $a0,xxx    # $a0 contiene il numero di byte richiesti.
                   # xxx deve essere multiplo di 4.
li      $v0,9      # codice 9 == allocazione memoria (sbrk)
syscall            # chiemete sistema.
                   # $v0 <-- contiene indirizzo del primo byte
                   # del blocco allocato dinamicamente
```

NB: non c'è modo di conoscere in anticipo il range di memoria che verrà restituito. L'unica garanzia è che si tratterà di un blocco contiguo della dimensione richiesta

Liste concatenate

Argomenti trattati:

i) Nodi della lista concatenata
ii) Allocazione memoria per un nodo
iii) Il nodo corrente
iv) Costruzione della lista concatenata mediante un ciclo
v) Stampa del contenuto di una lista concatenata

I nodi di una lista concatenata hanno questa struttura:

Ad eccezione del primo e dell'ultimo nodo della lista ogni nodo ha un **predecessore** ed un **successore** .

Una lista concatenata è uno dei (molti) modi di implementare il concetto di una lista ordinata. Ogni nodo contiene un dato e l'indirizzo del nodo successivo.

Liste concatenate

Le liste concatenate possono essere costruite in modo semplice in memoria statica. Ad esempio, per la seguente lista :

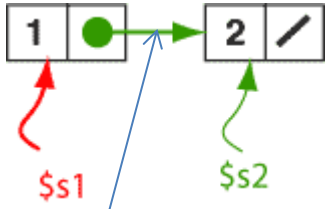
E' possibile una costruzione come quella mostrata di seguito:

```
.data

node01:     .word   1
            .word   node02

node02:     .word   2
            .word   0
```

Quanti byte servono per ogni nodo?
8 : 4 per l'intero e 4 per l'indirizzo.

Allocazione dinamica della memoria per un nodo

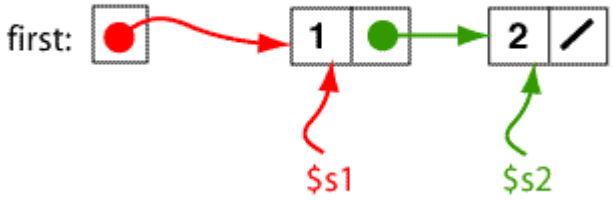
La memoria pe un nodo può essere allocata dinamicamente. La lista mostrata in figura può essere costruita come segue:




```
# primo nodo
li      $v0,9               # allocare
li      $a0,8               # 8 byte
syscall                     # $v0 <-- indirizzo
move    $s1,$v0             # $s1 = &(primo node)
li      $t0,1               # salva 1
sw      $t0,0($s1)          # a offset 0
                            # & è "indirizzo di"

# secondo
li      $v0,9               # allocare
li      $a0,8               # 8 byte
syscall                     # $v0 <-- indirizzo
move    $s2,$v0             # $s2 = &(secondo nodo)
                            # & è "indirizzo di"

# collegare secondo nodo al primo
sw    ..... ,4(.....)       # copia indirizzo secondo nodo
                            # nel primo nodo

# inizializzazione secondo nodo
li      $t0,2               # salva 2
sw      $t0,0($s2)          # a offset 0

# inserire valore nullo nel link del secondo nodo
sw    ..... ,4(.....)
```

Allocazione dinamica della memoria per un nodo

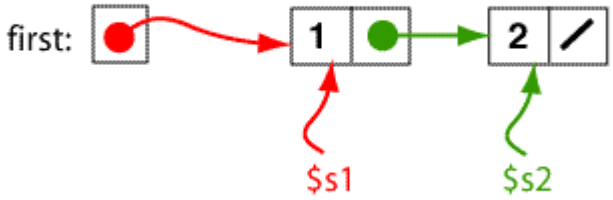
Quando una struttura dati (ad esempio la lista concatenata) è creata dinamicamente è prassi comune avere in memoria una variabile che punta al su




```
# primo nodo
        li      $v0,9               # allocare
        li      $a0,8               # 8 byte
        syscall                     # $v0 <-- indirizzo
        move    $s1,$v0             # $s1 = &(primo nodo)

        # copia dell'indirizzo in first
        sw      .....   ,   .....

        # inizializzazione del nodo
        li      $t0,1               # salva 1
        sw      $t0,0($s1)          # a offset 0

        # creazione secondo nodo...


.data

first: .word  0     # indirizo del primo nodo
```

Allocazione dinamica della memoria per un nodo

Quando una struttura dati (ad esempio la lista concatenata) è creata dinamicamente è prassi comune avere in memoria una variabile che punta al su



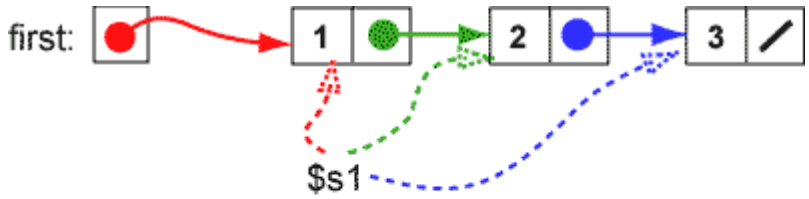

```
# primo nodo
        li      $v0,9               # allocare
        li      $a0,8               # 8 byte
        syscall                     # $v0 <-- indirizzo
        move    $s1,$v0             # $s1 = &(primo nodo)

        # copia dell'indirizzo in first
        sw      $s1 , first

        # inizializzazione del nodo
        li      $t0,1               # salva 1
        sw      $t0,0($s1)          # a offset 0

# creazione secondo nodo...


.data

first: .word  0     # indirizo del primo nodo
```

Lista dinamica composta da tre nodi

Costruzione di una lista concatenata di 3 nodi. Generalizzazione del processo di costruzione di una lista concatenata.

Il registro **$s1** punta al primo elemento
Il registro **$s1** punta al secondo elemento
Il registro **$s1** punta al terzo elemento
…

Perché usiamo sempre $s1 ? Se utilizzassimo un registro per ogni link ci ritroveremmo presto a corto di registri …



Lista dinamica composta da tre nodi

Costruzione primo nodo


```
main:
        # creazione primo nodo
        li      $v0,9               # allocare
        li      $a0,8               # 8 byte
        syscall                     # $v0 <-- indirizzo
        move    $s1,$v0             # $s1 = &(primo nodo)

        # copia indirizzo in first
        sw      .....   ,   .....

        # inizializzazione primo nodo
        li      $t0,1               # salva 1
        sw      $t0,0( ..... )      # a offset 0
```

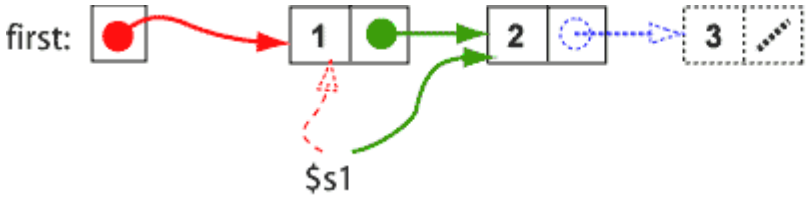
Lista dinamica composta da tre nodi

Costruzione **primo** nodo


```
main:
        # creazione primo nodo
        li      $v0,9               # allocare
        li      $a0,8               # 8 byte
        syscall                     # $v0 <-- indirizzo
        move    $s1,$v0             # $s1 = &(primo nodo)

        # copia indirizzo in first
        sw     $s1 , first

        # inizializzazione primo nodo
        li      $t0,1               # salva 1
        sw      $t0,0( $s1 )        # a offset 0
```

Lista dinamica composta da tre nodi

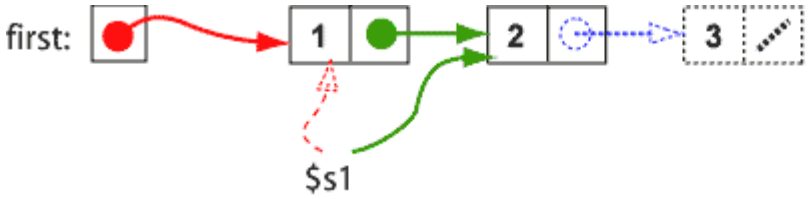
Costruzione **secondo** nodo




```
...

        # creazione secondo nodo
        li      $v0,9               # allocare
        li      $a0,8               # 8 byte
        syscall                     # $v0 <-- indirizzo

        # collegare il secondo nodo al primo

        sw      ..... ,4( ..... )  # copia indirizzo secondo nodo
                                    # nel primo

        # rendere il nuovo nodo il nodo corrente

        move    $s1,    .....       # $s1 = &(secondo nodo)

        # inizializzazione secondo nodo
        li      $t0,2               # salva 2
        sw      $t0,0($s1)          # a offset 0
```

Lista dinamica composta da tre nodi

Costruzione **secondo** nodo

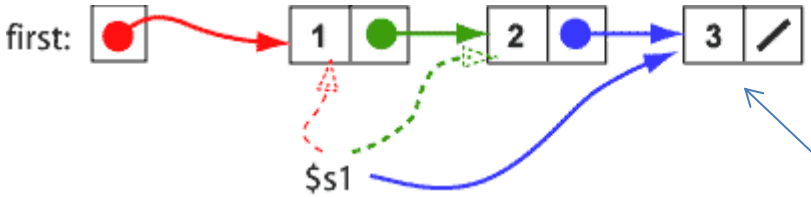


...

```
# creazione secondo nodo
li      $v0,9               # allocare
li      $a0,8               # 8 byte
syscall                     # $v0 <-- indirizzo

# collegare il secondo nodo al primo

sw     $v0 ,4( $s1 )   # copia indirizzo secondo nodo
                            # nel primo

# rendere il nuovo nodo il nodo corrente

move    $s1,    $v0       # $s1 = &(secondo nodo)

# inizializzazione secondo nodo
li      $t0,2               # salva 2
sw      $t0,0($s1)          # a offset 0
```

$s1 punta al nodo corrente

Costruzione **terzo** nodo



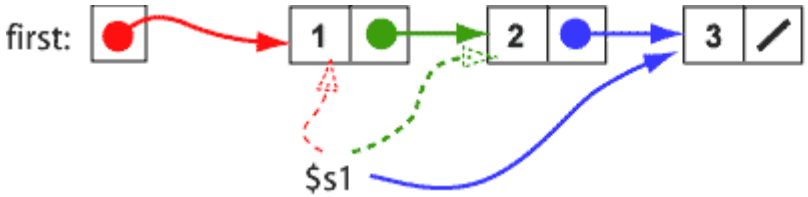
Il registro $s1 punta al nodo corrente… ma l'indirizzo del primo nodo non è andato perso poiché è salvato in **first**. Potremo sempre raggiungere l'inizio della lista.

Manca ancora il terzo nodo…

```
...
 # creazione terzo nodo
        li      $v0,9               # allocare
        li      $a0,8               # 8 byte
        syscall                     # $v0 <-- indirizzo

        # collegare il terzo nodo al secondo
        sw      $v0,4($s1)          # copia indirizzo del terzo nodo
                                    # nel secondo

        # Rendere il nuovo nodo il nodo corrente
        move    $s1,$v0             # $s1 = &(terzo nodo)

        # inizializzare il terzo nodo
        li      $t0,3               # salva 3
        sw      $t0,0($s1)          # a offset 0

        # chiusura della lista
        sw      $0,4($s1)           # inserire 0 nel link dell'ultimo nodo
```

$s1 punta al nodo corrente

Costruzione **terzo** nodo



Il cambiamento di significato di **$s1** è il **punto critico** del programma. Dobbiamo assicurarci di aver salvato il valore di $s1 nel secondo nodo **prima** di copiare in esso il valore contenuto in **$v0**.

```
...
 # creazione terzo nodo
        li      $v0,9               # allocare
        li      $a0,8               # 8 byte
        syscall                     # $v0 <-- indirizzo

    # collegare il terzo nodo al secondo
    sw      $v0,4($s1)              # copia indirizzo del terzo nodo
                                    # nel secondo

    # Rendere il nuovo nodo il nodo corrente
    move    $s1,$v0                 # $s1 = &(terzo nodo)

    # inizializzare il terzo nodo
    li      $t0,3                   # salva 3
    sw      $t0,0($s1)              # a offset 0

    # chiusura della lista
    sw      $0,4($s1)               # inserire 0 nel link dell'ultimo nodo
```

Programma completo

```
# listatrenodi.asm
#
        .text
        .globl  main

main:
        # creazione primo nodo
        li      $v0,9
        li      $a0,8
        syscall
        move    $s1,$v0

        # copia puntatore in first
        sw      $s1,first

        # inizializzazione primo nodo
        li      $t0,1
        sw      $t0,0($s1)

        # creazione secondo nodo
        li      $v0,9
        li      $a0,8
        syscall

        # collegare secondo nodo al primo
        sw      $v0,4($s1)


        # rendere secondo nodo il nodo corrente
        move    $s1,$v0

        # inizializzazione secondo nodo
        li      $t0,2
        sw      $t0,0($s1)

        # creazione terzo nodo
        li      $v0,9
        li      $a0,8
        syscall

        # collegare terzo nodo al secondo
        sw      $v0,4($s1)

        # rendere terzo nodo il nodo corrente
        move    $s1,$v0

        # inizializzazione terzo nodo
        li      $t0,3
        sw      $t0,0($s1)

        # chiusura lista
        sw      $0,4($s1)

        li      $v0,10
        syscall

        .data
first:  .word   0
```

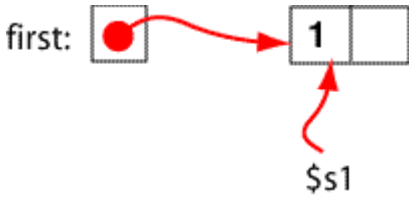
Domanda:

Come la creereste una lista concatenata composta da 100 nodi?

Creazione liste concatenate mediante cicli

Vogliamo costruire lista composta da 8 nodi. Il primo nodo lo creiamo esternamente al ciclo in modo da rendere più semplice il salvataggio del suo indirizzo in first:

```
main:
        # creazione primo nodo
        li      $v0,9               # allocare
        li      $a0,8               # 8 byte
        syscall                     # $v0 <-- indirizzo
        move    $s1,$v0             # $s1 = &(primo nodo)

        # copia indirizzo in first
        sw      $s1,first

        # inizializzazione del primo nodo
        li      $t0,1               # salva 1
        sw      $t0,0($s1)          # a offset 0
```



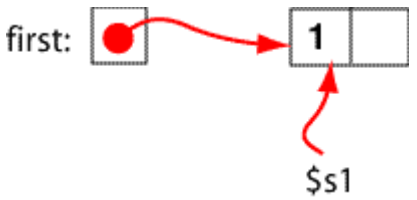
Ora possiamo concentrarci sul ciclo. Il contatore va da … a … ? Perché?

```
# creazione dei nodi rimanenti usando un ciclo
        li      $s2, .....      # contatore = ???
        li      $s3, .....      # limite_superiore
loop:   bgtu    ..... , ..... ,done     # while (contatore <= limite_superiore )
        # creazione di un nodo
          .  .  .  .  .
        addi    $s2,$s2,1       # contatore++
        b       loop
done:
        # chiusura lista
```

Creazione liste concatenate mediante cicli

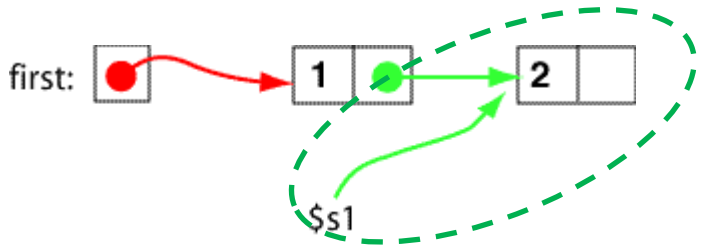
Vogliamo costruire lista composta da 8 nodi. Il primo nodo lo creiamo esternamente al ciclo in modo da rendere più semplice il salvataggio del suo indirizzo in first:

```
main:
        # creazione primo nodo
        li      $v0,9               # allocare
        li      $a0,8               # 8 byte
        syscall                     # $v0 <-- indirizzo
        move    $s1,$v0             # $s1 = &(primo nodo)

        # copia indirizzo in first
        sw      $s1,first

        # inizializzazione del primo nodo
        li      $t0,1               # salva 1
        sw      $t0,0($s1)          # a offset 0
```

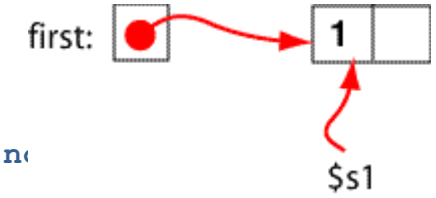


Ora possiamo concentrarci sul **ciclo**. Il contatore va da … a … ? Perché?

```
# creazione dei nodi rimanenti usando un ciclo
        li      $s2, 2              # contatore = ???
        li      $s3, 8              # limite_superiore
loop:   bgtu    $s2 , $s3 ,done     # while (contatore <= limite_superiore )
        # creazione di un nodo
          .  .  .  .  .
        addi    $s2,$s2,1           # contatore++
        b       loop
done:
        # chiusura lista
```

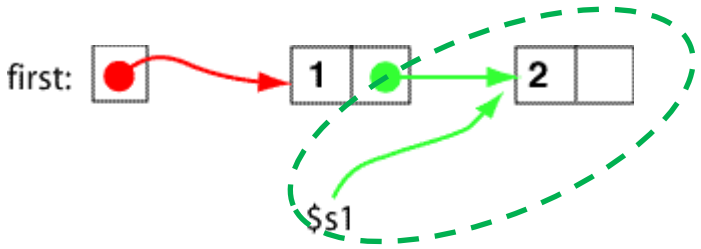


Creazione liste concatenate mediante cicli

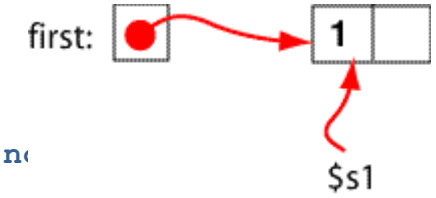


Codice completo del ciclo:

```
#
# All'ingresso nel ciclo:  $s1 punta al primo n
#
loop:   bgtu    $s2,$s3,done      # while (contatore <= limite_superiore )

        # creazione di un nodo
        li      $v0,9             # allocare
        li      $a0,8             # 8 byte
        syscall                   # $v0 <-- indirizzo

        # collegamento di questo nodo al precedente
                                  # $s1 = &(nodo precedente)
        sw      $v0,4($s1)        # copia indirizzo nodo corrente
                                  # nel precedente

        # rendere nodo appena creato nodo corrente
        move    $s1,$v0

        # inizializzazione dle nuovo nodo
        sw      $s2,0($s1)        # salva valore contatore
                                  # come valore del nodo corrente

        addi    $s2,$s2,1         # contatore++
        b       loop

done:
```

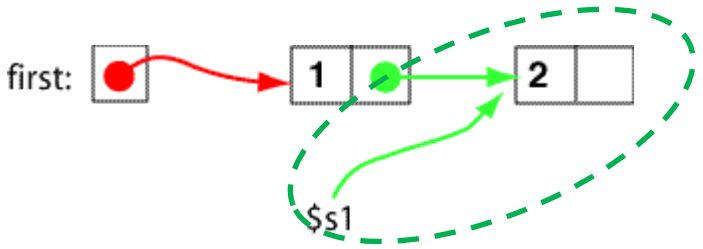


Creazione liste concatenate mediante cicli



Codice completo del ciclo:

```
#
# All'ingresso nel ciclo:  $s1 punta al primo n
#
loop:   bgtu    $s2,$s3,done      # while (contatore <= limite_superiore )

        # creazione di un nodo
        li      $v0,9             # allocare
        li      $a0,8             # 8 byte
        syscall                   # $v0 <-- indirizzo

        # collegamento di questo nodo al precedente
                                  # $s1 = &(nodo precedente)
        sw      $v0,4($s1)        # copia indirizzo nodo corrente
                                  # nel precedente

        # rendere nodo appena creato nodo corrente
        move    $s1,$v0

        # inizializzazione dle nuovo nodo
        sw      $s2,0($s1)        # salva valore contatore
                                  # come valore del nodo corrente

        addi    $s2,$s2,1         # contatore++
        b       loop

done:
```



Qui dobbiamo chiudere la lista …

Domanda: All'inizio della prima iterazione del ciclo chi è il nodo corrente? E alla fine?

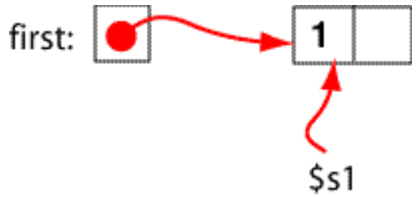
Creazione liste concatenate mediante cicli

Chiusura lista post completamento ciclo di costruzione:

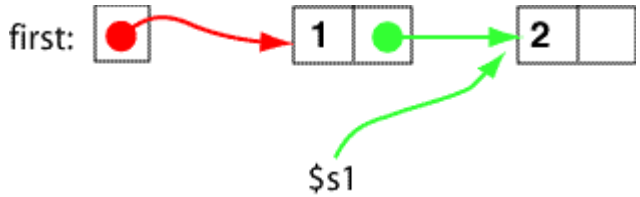
```
done:
        # chiusura della lista
        sw      $0,4($s1)          # inserire 0 nel campo link
                                   # del nodo corrente che è
                                   # anche l'ultimo nodo.
```

Creazione liste concatenate mediante cicli

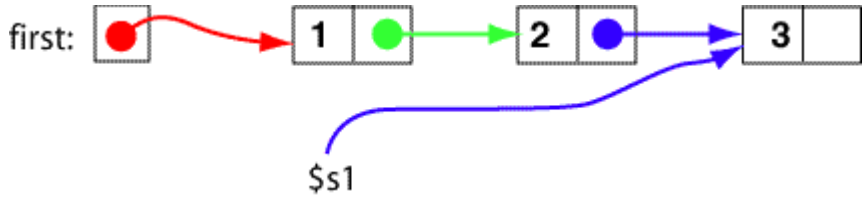
Prima dell'ingresso nel ciclo:



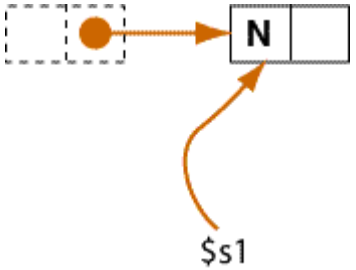
Dopo la prima iterazione:



Dopo la seconda iterazione:



In generale il nodo <u>corrente</u> può essere rappresentato così: **$s1** punta al nodo corrente ed esso è già **puntato anche dal suo predecessore**



Esercizio 1

Costruire un programma che chieda all'utente un numero N e costruisca una lista concatenata di N nodi.

Attraversamento della lista concatenata

Dopo la costruzione della lista l'indirizzo del suo primo nodo è contenuto in first. Vogliamo attraversare la lista e stampare i dati contenuti all'interno dei suoi nodi.

```
# Assume che la lista concatenata sia stata costruita,
# e che first punti al suo primo nodo.

          ...  .....  , .....   # Ottenere un puntatore al primo nodo
                                # inserire l'indirizzo in esso contenuto in $s0

lp:       beqz   $s0,endlp      # fino a quando il puntatore non è nullo
          lw     $a0,0($s0)     # estrarre il dato da questo nodo

                                #
          li     $v0,1          #   stampare il dato appena estra
          syscall               #
          la     $a0,sep        #   stampare un separatore
          li     $v0,4          #
          syscall               #

          lw     $s0,4($s0)     #   ottenere un puntatore al prossimo nodo
          b      lp

endlp:     . . .

          .data
first:    .word  0
sep:      .asciiz "  "
```

Attraversamento della lista concatenata

Dopo la costruzione della lista l'indirizzo del suo primo nodo è contenuto in first. Vogliamo attraversare la lista e stampare i dati contenuti all'interno dei suoi nodi.

```
# Assume che la lista concatenata sia stata costruita,
# e che first punti al suo primo nodo.

          lw  $s0 ,first        # Ottenere un puntatore al primo nodo
                                # inserire l'indirizzo in esso contenuto in $s0

lp:       beqz   $s0,endlp      # fino a quando il puntatore non è nullo
          lw     $a0,0($s0)     # estrarre il dato da questo nodo

                                #
          li     $v0,1          #   stampare il dato appena estra
          syscall               #
          la     $a0,sep        #   stampare un separatore
          li     $v0,4          #
          syscall               #

          lw     $s0,4($s0)     #   ottenere un puntatore al prossimo nodo
          b      lp

endlp:     . . .

          .data
first:    .word  0
sep:      .asciiz "   "
```

branch if equal to zero (beqz)

Programma completo

```
# listaottonodi.asm
.text
.globl  main

main:
        # creare la lista concatenata

        # $s1 nodo corrente in ciclo costruzione lista
        # $s2 contatore ciclo costruzione

        # creazione primo nodo
        li      $v0,9
        li      $a0,8
        syscall
        move    $s1,$v0

        # backup puntatore a primo nodo
        sw      $s1,first

        # inizializzazione primo nodo
        li      $t0,1
        sw      $t0,0($s1)

        # creazione nodi rimanenti
        li      $s2,2
        li      $s3,8

loop:   bgtu    $s2,$s3,done

        # creazione di un nodo
        li      $v0,9
        li      $a0,8
        syscall

        # collegamento del nodo al precedente

        sw      $v0,4($s1)


        # rendere nodo creato nodo corrente
        move    $s1,$v0

# inizializzazione nodo
        sw      $s2,0($s1)

        addi    $s2,$s2,1
        b       loop

done:
        # chiusura lista
        sw      $0,4($s1)



        # stampa contenuto lista
        # $s0 nodo corrente nel ciclo di stampa

        lw      $s0,first           # ottenere puntatore a primo nodo
lp:     beqz    $s0,endlp           # fino a quando il puntatore non
è nullo
        lw      $a0,0($s0)          # leggi dato nodo
        li      $v0,1               # stampa
        syscall                     #
        la      $a0,sep             # stampa separatore
        li      $v0,4               #
        syscall                     #

        lw      $s0,4($s0)          # ottieni puntatore a prossimo
nodo
        b       lp

endlp:
        li      $v0,10
        syscall

       .data
first: .word   0
sep:   .asciiz "  "
```

Esercizio 2

Costruire un programma che chieda all'utente un numero N e costruisca una lista concatenata di N nodi. Dopo la costruzione della lista il programma chiede all'utente l'indice di un nodo nella lista concatenata e stampa i nodi della lista a partire dall'indice inserito dall'utente.

Esercizio 3

Costruire un programma che chieda all'utente un numero N e costruisca una lista concatenata di N nodi.

Ogni nodo contiene un record composto dai seguenti campi:

GIOCATORE_id | SQUADRA_id | PUNTEGGIO | NUM_PARTITE GIOCATE

oltre al puntatore all'eventuale nodo successivo nella lista. Durante la creazione di ogni nodo il programma chiederà all'utente di inserire i dati necessari per la compilazione del record contenuto in esso.

Dopo aver costruito la lista concatenata la si attraversi e si stampi il valore medio di NUM_PARTITEGIOCATE