PROGETTO ARCHITETTURE

CASULA CARMELO

7085019

Per la creazione della lista mi sono appoggiato alla memoria facendo in modo che all'interno del byte 0 ci fosse l'elemento da inserire nella catena, mentre nei restanti 4 byte c'è l'indirizzo dell'elemento successivo (ovviamente nella coda c'è l'indirizzo della testa)

ESEMPIO CON ALCUNI ELEMENTI INSERITI VISUALIZZATI IN ASCII PER VEDERE L'ELEMENTO:

0x0000051c			X	X	X
0x00000518					
0x00000514	♣E	E		♣	
0x00000510	♣¶	¶	♣		
0x0000050c	d♣	♣			d
0x00000508	¤c			c	¤
0x00000504	♣...		B	...	♣
0x00000500	♣♣a	a	♣	♣	

ESEMPIO CON ALCUNI ELEMENTI INSERITI VISUALIZZATI IN ESADECIMALE PER VEDERE L'INDIRIZZO:

0x0000051c	0x00000000	0x00	X	X	X
0x00000518	0x00000000	0x00	0x00	0x00	0x00
0x00000514	0x00050045	0x45	0x00	0x05	0x00
0x00000510	0x00000514	0x14	0x05	0x00	0x00
0x0000050c	0x64000005	0x05	0x00	0x00	0x64
0x00000508	0x0f630000	0x00	0x00	0x63	0x0f
0x00000504	0x050a4200	0x00	0x42	0x0a	0x05
0x00000500	0x00050561	0x61	0x05	0x05	0x00

Per verificare che l'input del comando sia stato scritto in modo corretto si caricano prima i vari caratteri da verificare nei registri T, poi scorrendo di un byte alla volta la stringa e salvando il carattere correte su t0 si va a verificare tramite bne (branch if not equal) se il carattere corrisponde. Se la prima lettera non corrisponde si salta all'istruzione successiva e si verifica se corrisponde con quella determinata istruzione e se anche la seconda lettera non corrisponde si passa a end_comando che si occupa di scorrere la lista finché non trova il carattere separatore "~" tornando all'inizio del programma e facendo ripartire tutto; se il corpo del comando principale è stato scritto bene si salta a una procedura unica per tutti che verifica che effettivamente tra il comando e il carattere separatore non ci sia nient'altro come ad esempio altri comandi, caratteri a caso e altre cose, in questa procedura semplicemente viene salvato il punto in cui si è nella stringa la si scorre, se viene trovato il carattere separatore o se si arriva alla fine della stringa torna ad eseguire il comando perché scritto bene

altrimenti scorre la stringa e se trova degli elementi che non siano lo spazio vuoto o ~ scorre finché non trova l'inizio del prossimo comando o la fine.

Ogni volta che si esegue un comando(che abbia successo o no) si aumenta il contatore dei comandi e si verifica se si è raggiunto il numero massimo, in questo caso si è scelto di mettere un massimo di 30 che una volta raggiunto terminerà automaticamente il programma, nel caso in cui dentro end_comando si arriverà alla fine della lista il programma terminerà lo stesso . Sottolineammo che end_comando può essere raggiunto anche quando l'istruzione è stata eseguita con successo comportandosi sempre allo stesso modo.

L'istruzione ADD viene gestita nel seguente modo:

-Prima di tutto come per ogni istruzione si verifica come spiegato precedentemente se è stato scritto nel modo corretto;

-Successivamente si va a verificare se i prossimi 5 byte che si hanno a disposizione sono liberi, per questo codice si è scelto come indirizzo di partenza l'indirizzo "0x00000800" salvato sia in s3 che verrà usato per scorrere la memoria mano a mano che si aggiungono nuovi elementi, sia in s8 dove viene modificato solo in caso di cancellazioni, ma questo sarà spiegato successivamente quando si parlerà dell'operazione DEL; prima di tutto viene caricato l'elemento nel primo byte disponibile e dal secondo byte in poi lo faccio puntare alla testa, se sarà il primo elemento inserito andrà a puntare a se stesso, in t1 salvo l'indirizzo del nodo appena inserito per fare in modo che il nodo precedente punti al nuovo nodo per fare ciò mano a mano che si va avanti inserendo salvo l'indirizzo del nodo inserito in s7(all'inizio ci sarà dentro la testa) modificando così il puntatore al prossimo elemento, nel caso sia il primo elemento punterà sempre a se stesso dato che s7 viene aggiornato dopo che ho fatto questa procedura(in pratica all'interno di s7 viene memorizzata la coda) e infine aumento s3 di 5 per andare a vedere se ci sarà una nuova cella disponibile nel caso di una nuova aggiunta;

I registri temporanei usati sono: t0 per il carattere da inserire, da t1 a t4 per verificare se il comando è scritto in modo corretto, e successivamente t1 viene utilizzato per memorizzare l'indirizzo della nuova coda per fare in modo che la vecchia coda punti alla nuova e non più alla testa

IMPLEMENTAZIONE DI ADD:

```
 sb t0, 0(s3)
 addi s0, s0 1

sw s8,1(s3)

add t1 zero s3

sw t1, 1(s7)
add s7 zero s3
addi s3, s3 5

 j end_comando
```

Istruzione DEL:

Come per ogni comando controllo allo stesso modo se è stato scritto correttamente, e successivamente salvo su t1 l'indirizzo della testa, t2 lo utilizzo per scorrere gli elementi partendo dalla testa e in t3 salvo l'elemento da cercare per verificare se l'elemento che deve essere cancellato è presente nella cella che si sta osservando in quel momento, Prima di tutto verifico se esiste la lista guardando se nell'indirizzo puntato da s8(indirizzo della testa) c'è effettivamente qualcosa se non c'è nulla vuol dire che la lista non esiste e quindi termino il programma, successivamente controllo pure se l'elemento punta a se stesso, in quel caso verifico se contiene l'elemento se lo contiene azzero il tutto altrimenti termino il comando; Successivamente ho implementato due diverse procedure, una che controlla prima di tutto se l'elemento da eliminare è dentro la testa se si prosegue prima di tutto a far diventare zero la prima cella e carico su t4 l'indirizzo a cui punta tale nodo per renderlo la nuova testa e successivamente azzerando pure il puntatore, pongo il controllore della testa al nuovo nodo e torno indietro all'inizio di questa procedura per verificare che nella nuova testa non ci sia sempre lo stesso elemento, in caso contrario si passa alla seconda procedura rendendo irraggiungibile la prima salvando di nuovo il carattere in t3 e verifico se è presente nella cella se si azzero come prima la cella poi vado all'elemento precedente che è stato salvato in t5(più avanti sarà spiegato come), e la faccio puntare all'indirizzo a cui puntava la cella eliminata (l'indirizzo è stato salvato in t4), cancello anche

il puntatore nel vecchio nodo e vado a verificare se sono arrivato alla fine se si passo a fine cancellazioni altrimenti verifico se nel successivo bisogna eliminare in questo caso supponiamo che non bisogni cancellare nulla, si andrà avanti e si salverà quell'indirizzo in t5 nel caso si elimini il successivo elemento, si riverifica se si è arrivati alla fine come prima se si si prova a caricare t6 che punta alla testa sia che sia stata modificata e punti a una nuova sia se punta sempre alla vecchia nella coda e la si reinserisce pure in s8 per sicurezza nel caso sia stata modificata;

I registri temporanei usati sono t0 per memorizzare il carattere da eliminare, da t1 a t5 prima per verificare se il comando è stato scritto correttamente, successivamente t1 per memorizzare la testa, t2 per scorrere la lista, t3 per caricare il carattere nella cella da confrontare, t4 per salvare l'indirizzo successivo e t5 per il precedente, infine t6 viene utilizzato per avere un registro con la testa che non deve essere modificato;

MEMORIA IN ASCII E IN ESADECIMALE PRIMA DI ELIMINARE IL CARATTERE B:

0x00000514					
0x00000510	♣		♣		
0x0000050c	d♣	♣			d
0x00000508	¤c			c	¤
0x00000504	♣…		B	…	♣
0x00000500	♣♣a	a	♣	♣	

0x00000514	0x00000000	0x00	0x00	0x00	0x00
0x00000510	0x00000500	0x00	0x05	0x00	0x00
0x0000050c	0x64000005	0x05	0x00	0x00	0x64
0x00000508	0x0f630000	0x00	0x00	0x63	0x0f
0x00000504	0x050a4200	0x00	0x42	0x0a	0x05
0x00000500	0x00050561	0x61	0x05	0x05	0x00

MEMORIA IN ASCII E IN ESADECIMALE PRIMA DOPO AVER ELIMINATO IL CARATTERE B:

0x00000514	♣E	E		♣	
0x00000510	♣¶	¶	♣		
0x0000050c	d♣	♣			d
0x00000508	¤c			c	¤
0x00000504					
0x00000500	♣...	a	...	♣	

0x0000051c	0x00000000	0x00	X	X	X
0x00000518	0x00000000	0x00	0x00	0x00	0x00
0x00000514	0x00050045	0x45	0x00	0x05	0x00
0x00000510	0x00000514	0x14	0x05	0x00	0x00
0x0000050c	0x64000005	0x05	0x00	0x00	0x64
0x00000508	0x0f630000	0x00	0x00	0x63	0x0f
0x00000504	0x00000000	0x00	0x00	0x00	0x00
0x00000500	0x00050a61	0x61	0x0a	0x05	0x00

Istruzione PRINT:

Per l'operazione PRINT carico la testa su t0 per scorrere la lista, successivamente carico il carattere puntato da t0 su t1 lo sommo con 0 dentro a0 e lo stampo a video, scorro la lista e verifico se l'ultimo elemento stampato punta alla testa contenuta sempre in s8 se si è fatto il giro completo della lista si finisce l'operazione come tutte le altre in caso contrario si continua a stampare finché non si arriva alla testa, una volta arrivati alla testa si va a capo per fare in modo che se si fanno altre stampe non si sovrappongano; In questa procedura non si fa nessuna modifica in memoria ne ai puntatori ne agli elementi ; I registri temporanei utilizzati oltre ai soliti per verificare che il comando sia scritto correttamente sono t0 per scorrere la lista e t1 per salvare il carattere da stampare

IMPLEMENTAZIONE E STAMPA:

```
167
168     add t0  s8  zero
169     stampa:
170     lb t1 0(t0)
171     add a0 zero t1
172     li a7 11
173     ecall
174     addi t0 t0 1
175     lw t0,0(t0)
176     bne t0 s8 stampa
177     la a0 newline
178     li a7 4
179     ecall
```

Console

aBcdE

Istruzione SDX:

Per fare lo scorrimento verso sinistra mi appoggio agl'indici, carico prima la testa su t0, poi il primo elemento su t1 e l'indirizzo del secondo elemento su t2, salvo poi su t5 l'elemento che deve essere sostituito , lo modifico con quello contenuto su t1, salvo l'indirizzo del prossimo elemento su t2 come prima e l'elemento sostituito in t1, controllo se sono arrivato all'ultimo elemento se si aggiorno pure la testa con il nuovo elemento(quello che era nella coda all'inizio) e faccio finire l'operazione; gli indici restano invariati;

i registri temporanei usati sono t0 per la testa, t1 per l'elemento da spostare, t2 per l'indirizzo da raggiungere e t5 per l'elemento sostituito;

MEMORIA DOPO LO SHIFT:

0x0000051c			X	X	X
0x00000518					
0x00000514	♣d	d		♣	
0x00000510	♣¶	¶	♣		
0x0000050c	c♣	♣			c
0x00000508	¤B			B	¤
0x00000504	♣…		a	…	♣
0x00000500	♣♣E	E	♣	♣	

IMPLEMENTAZIONE E STAMPA:

```
215     lb t1, 0(t0)
216     lw t2 1(t0)
217     back:
218     lb t5, 0(t2)
219
220     sb t1, 0(t2)
221     lw t2, 1(t2)
222     add t1 zero t5
223     bne t2 s8 back
224     sb t1, 0(t2)
225     j end_comando
226 ope_SSX:
```

Console

aBcdE
EaBcd

Istruzione SSX:

Per questa operazione salvo prima il contenuto della testa su t0, e in t1 l'indirizzo della testa, lo copio pure in t2, aggiorno t1 con l'indirizzo dell'elemento a destra e successivamente salvo su t3 l'elemento di quella cella per poi caricarlo nella cella precedente(puntata da t2) verifico se sono arrivato alla fine della lista, se non ci sono arrivato ripeto finche non ci arrivo, una volta arrivato alla fine salvo il primo elemento(salvato in t0) nella coda puntata da t2; I registri temporanei utilizzati sono t0 per il primo elemento t1 per scorrere la lista, t2 per salvare l'indirizzo dove sostituire l'elemento e t3 per salvare l'elemento da spostare

MEMORIA DOPO LO SHIFT:

0x0000051c			X	X	X
0x00000518					
0x00000514	♣a	a		♣	
0x00000510	♣¶	¶	♣		
0x0000050c	E♣	♣			E
0x00000508	¤d			d	¤
0x00000504	♣…		c	…	♣
0x00000500	♣♣B	B	♣	♣	

IMPLEMENTAZIONE E STAMPA:

```
371    add t1 zero(s8)
372    lb t0, 0(s8)
373    loop:
374    add t2 zero t1
375    lw t1 1(t1)
376    lb t3 0(t1)
377    sb t3 0(t2)
378    bne t1 s8 loop
379    sb t0 0(t2)
380    j end comando
```

Console

```
aBcdE
BcdEa
```

Istruzione REV:

Per l'istruzione di reversione della lista memorizzo 4 volte la testa in t0 per tenerla salvata per prendere il primo elemento, in t1 e t3 per scorrere la lista e in t4 per vedere se si è arrivati alla fine; per trovare gli elementi da invertire prima si scorre la lista memorizzando prima di andare avanti di una casella in t1 l'indirizzo attuale e inserendo in t3 il prossimo indirizzo e verificando se corrisponde con t4 dove la prima volta c'è l'indirizzo della testa, successivamente c'è l'indirizzo dell'ultimo elemento scambiato; una volta che si è trovato l'indirizzo dell'elemento più in fondo da scambiare si salva in t5 l'elemento che viene prima e in t6 l'elemento che viene dopo, per poi caricare t6 nell'indirizzo puntato da t0 ovvero la cella che viene prima e t5 nella cella puntata da t1 ovvero quella che viene dopo, una volta fatto questo si aggiornato t0 con la cella successiva, t1 e t3 con la cella dalla quale bisogna partire e t4 con la cella dove bisogna fermarsi e verifico se l'indirizzo contenuto in t1 è più piccolo dell'indirizzo contenuto in t4 nel caso siano uguali o t4 sia più piccolo vuol dire che gli elementi sono stati tutti invertiti e si può terminare l'operazione.

MEMORIA DOPO LA REVERSIONE:

0x00000514	♣a	a		♣	
0x00000510	♣¶	¶	♣		
0x0000050c	B♣	♣			B
0x00000508	¤c			c	¤
0x00000504	♣...		d	...	♣
0x00000500	♣♣E	E	♣	♣	

IMPLEMENTAZIONE E STAMPA:

```
299     return:
300         add t1 zero t3
301         lw t3 1(t1)
302         bne t3 t4 return
303         lb t5 0(t0)
304         lb t6 0(t1)
305         sb t6 0(t0)
306         sb t5 0(t1)
307         lw t0 1(t0)
308         add t4 zero t1
309         add t1 zero t0
310         add t3 zero t0
311         blt t1 t4 return
312         j end_comando
313 end_comando:
314
```

Console

```
aBcdE
EdcBa
```

Istruzione SORT

Per questa istruzione divido momentaneamente la lista in quattro sotto liste che verranno ognuna ordinata singolarmente per poi riunire tutto in un'unica lista dal momento che la lista deve essere ordinata secondo il criterio punteggiatura-numeri-minuscole-maiuscole,prima scorro la lista completa cercando gli elementi che hanno codice ASCII corrispondente al tipo che ci serve, prendiamo per esempio in considerazione la punteggiatura, l'elemento viene individuato tramite dei salti condizionati che fanno avanzare lo scorrimento della lista (per gli elementi di

punteggiatura non è necessario modificare il puntatore della coda dato che sono i primi ma dai numeri in poi viene modificato con l'indirizzo del primo elemento dove si va a cercare, l'indirizzo della coda viene trovato prima di ogni cosa tramite l'apposita procedura), non appena si trova un elemento di punteggiatura lo si mette nella posizione più vicina alla testa senza dentro caratteri di punteggiatura, questa posizione è controllata dal registro t4 che ogni volta che viene intercettato il carattere prima memorizza la sua vecchia posizione in t6 per poi andare a puntare alla cella successiva una volta che si ha esplorato tutta la lista si verifica se si sono trovati gli elementi desiderati confrontando t4 e t6 che all'inizio contengono entrambi l'indirizzo della testa(o se si sta radunando altri tipi di caratteri conterranno la testa della sotto lista momentanea) e quindi se non si è trovato nessun elemento si va avanti per il prossimo tipo di carattere altrimenti si prosegue per fare l'ordinamento caricando in t0 la testa della sotto-lista e in t2 l'indirizzo del secondo elemento, però se t1 e t2 coincidono vuol dire che c'è solo un elemento nella sotto lista e quindi si può proseguire al successivo tipo di carattere, per l'ordinamento prima faccio puntare l'ultimo elemento al primo della sotto lista(nel caso degli elementi di punteggiatura alla testa della lista per gli altri elementi salvo in s10 il primo elemento della sotto lista e dopo che ho radunato tutti quei determinati caratteri faccio puntare l'ultimo all'indirizzo contenuto dentro s10) successivamente ordino con bubble sort ovvero confronto l'elemento a sinistra con quello a destra e se quello a sinistra è più grande gli scambio e lo confronto con il successivo, se è più grande quello a destra lo confronto con il suo vicino di destra finche non arrivo alla fine, una volta arrivato alla fine scorro di nuovo tutta la lista confrontando gli elementi e verificando che siano in ordine crescente se ne trovo due che non sono ordinati torno a fare gli scambi, una volta ordinata la sotto lista faccio puntare di nuovo la coda al suo vero elemento successore nella vera lista e controllo se l'ultimo elemento della famigerata sotto lista punta alla testa della lista completa se si vuol dire che non ci sono altri caratteri e posso terminare il comando altrimenti proseguo nello stesso modo ad

ordinare gli altri elementi eccezion fatta per le maiuscole che dato sono l'ultimo tipo di carattere da ordinare non c'è bisogno di accorparle e si può subito ordinarle;

Per questa operazione vengono usati tutti i registri t più s10 utilizzato per memorizzare ogni volta che si crea una sotto lista l'indirizzo a cui puntava prima la coda della lista temporanea,

t0 e t2 vengono utilizzati per scorrere la lista sia quando si cercano i caratteri sia quando si ordina

t1 viene utilizzato prima per salvare i caratteri ASCII da confrontare sia come registro di scambio per l'ordinamento

t3 viene utilizzato sia per salvare il carattere corrente nella ricerca sia come registro di scambio durante l'ordinamento

t4 è utilizzato per le operazioni di ricerca

t5 è usato sia come registro di scambio nella ricerca sia per controllore durante l'ordinamento

t6 è usato come registro di scambio nella ricerca e per le memorizzazioni degli indirizzi della sotto lista(queste operazioni non andranno mai in conflitto perché t6 viene si modificato per scambiare i caratteri ma subito dopo viene riaggiornato con l'indirizzo che serve)

MEMORIA PRIMA E DOPO ORDINAMENTO

0x00000738	•		•		
0x00000734	3•	•			3
0x00000730	7!			!	7
0x0000072c	•2z		z	2	•
0x00000728	•-T	T	-	•	
0x00000724	•(	(	•		
0x00000720	W•	•			W
0x0000071c	#t			t	#
0x00000718	•▲t		t	▲	•
0x00000714	•↓8	8	↓	•	
0x00000710	•¶	¶	•		
0x0000070c	1•	•			1
0x00000708	¤B			B	¤
0x00000704	•…		;	…	•
0x00000700	•♣a	a	♣	•	

0x00000734	W•	•			W
0x00000730	7T			T	7
0x0000072c	•2B		B	2	•
0x00000728	•-z	z	-	•	
0x00000724	•(	(	•		
0x00000720	t•	•			t
0x0000071c	#t			t	#
0x00000718	•▲a		a	▲	•
0x00000714	•↓8	8	↓	•	
0x00000710	•¶	¶	•		
0x0000070c	3•	•			3
0x00000708	¤1			1	¤
0x00000704	•…		;	…	•
0x00000700	•♣!	!	♣	•	

ALCUNE STRINGHE TESTATE PRESENTI ALL'INTERNO DEL CODICE

"ADD(1) ~ ADD(a) ~ ADD(a) ~ ADD(B) ~ ADD(;) ~ ADD(9) ~SSX~SORT~PRINT~DEL(b)~DEL(B)

~PRI~SDX~REV~PRINT"

"ADD(1) ~ SSX ~ ADD(a) ~ add(B) ~ ADD(B) ~ ADD ~ ADD(9)
~PRINT~SORT(a)~PRINT~DEL(bb)~DEL(B) ~PRINT~REV~SDX~PRINT"

"ADD(a)~ADD(;)~ADD(B)~ADD(1)~ADD(8)~ADD(t)~ADD(t)~ADD(W)~ADD(T)ADD(z)~ADD(!)~ADD(3)
~SORT~PRINT~ADD(W)~PRINT"
"ADD(a)~ADD(d)~ADD(a)~ADD(f)~SSX~PRINT"
"ADD(a)~ADD(B)~ADD(c)~ADD(d)~ADD(E)~PRINT~SSX~PRINT"