

Progetto Assembly RISC-V per il Corso di Architetture degli Elaboratori – A.A. 2024/2025 – Gestione di Liste Concatenate

Versione Iniziale del documento, aggiornata il 03/04/2025.

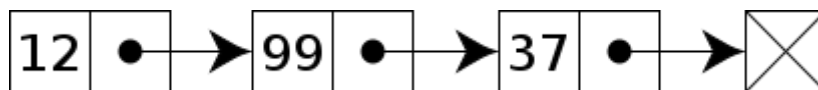
-

Liste Concatenate

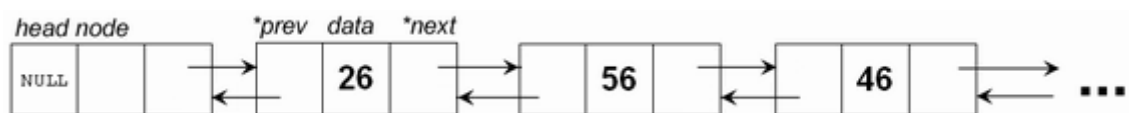
Una lista concatenata (o Linked List) è una struttura dati dinamica che consiste di una sequenza di nodi, ognuno contenente campi di dati arbitrari ed uno o due riferimenti ("link") che puntano al nodo successivo e/o precedente. Le liste concatenate permettono l'inserzione e la rimozione di nodi in ogni punto della lista, ma – diversamente dagli array – non permettono l'accesso casuale, solo quello sequenziale.

Esistono diversi tipi di liste concatenate (immagini da wikipedia):

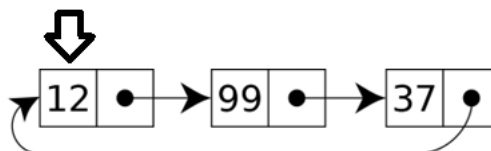
- liste concatenate semplici,



- liste concatenate doppie e



- liste circolari



Liste Concatenate Semplici in RISC-V

Il progetto di AE 24-25 mira all'implementazione di un codice RISC-V che gestisce alcune delle operazioni fondamentali per una lista concatenata semplice, tra le quali:

- ADD - Inserimento di un elemento
- DEL - Rimozione di un elemento

- PRINT - Stampa della lista
- SORT - Ordinamento della lista
- SDX – Shift a destra (rotazione in senso orario) degli elementi della lista
- SSX – Shift a sinistra (rotazione in senso antiorario) degli elementi della lista
- REV – Inversione degli elementi della lista

Ogni elemento della lista si deve supporre di dimensione 5 byte, così suddivisi:

- DATA(Byte 0): contiene l'informazione
- PAHEAD (Byte 1-4): puntatore all'elemento successivo, o a 0x00000000 se unico o ultimo elemento della lista

Si noti come i puntatori alla memoria abbiamo **dimensione 32bit, ovvero una word RISC-V**. Inoltre, si assume che il byte di informazione contenuto in ciascun elemento della lista rappresenti un carattere ASCII. Sui caratteri ASCII viene stabilito il seguente ordinamento (transitivo):

- una lettera maiuscola (ASCII da 65 a 90 compresi) viene sempre ritenuta maggiore di una minuscola
- una lettera minuscola (ASCII da 97 a 122 compresi) viene sempre ritenuta maggiore di un numero
- un numero (ASCII da 48 a 57 compresi) viene sempre ritenuto maggiore di un carattere extra che non sia lettera o numero
- non si considerano accettabili caratteri extra con codice ASCII minore di 32 o maggiore di 125.

All'interno di ogni categoria vige l'ordinamento dato dal codice ASCII. Per esempio, date due lettere maiuscole x e x' , $x < x'$ se e solo se $\text{ASCII}(x) < \text{ASCII}(x')$. Lo stesso vale per le lettere minuscole, per i numeri e per i caratteri extra. Ad esempio, la sequenza a.2Er4,w si ordinerà come ,.24arwE

Sotto si riassumono i **codici ASCII accettabili (da 32 a 125 compresi)** come informazioni negli elementi della lista, per questo progetto.

Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
32	20	040	0x32	Space	64	40	100	0x64	@	96	60	140	0x96	`
33	21	041	0x33	!	65	41	101	0x65	A	97	61	141	0x97	a
34	22	042	0x34	"	66	42	102	0x66	B	98	62	142	0x98	b
35	23	043	0x35	#	67	43	103	0x67	C	99	63	143	0x99	c
36	24	044	0x36	\$	68	44	104	0x68	D	100	64	144	0x100	d
37	25	045	0x37	%	69	45	105	0x69	E	101	65	145	0x101	e
38	26	046	0x38	&	70	46	106	0x70	F	102	66	146	0x102	f
39	27	047	0x39	'	71	47	107	0x71	G	103	67	147	0x103	g
40	28	050	0x40	(72	48	110	0x72	H	104	68	150	0x104	h
41	29	051	0x41)	73	49	111	0x73	I	105	69	151	0x105	i
42	2A	052	0x42	*	74	4A	112	0x74	J	106	6A	152	0x106	j
43	2B	053	0x43	+	75	4B	113	0x75	K	107	6B	153	0x107	k
44	2C	054	0x44	,	76	4C	114	0x76	L	108	6C	154	0x108	l
45	2D	055	0x45	-	77	4D	115	0x77	M	109	6D	155	0x109	m
46	2E	056	0x46	.	78	4E	116	0x78	N	110	6E	156	0x110	n
47	2F	057	0x47	/	79	4F	117	0x79	O	111	6F	157	0x111	o
48	30	060	0x48	0	80	50	120	0x80	P	112	70	160	0x112	p
49	31	061	0x49	1	81	51	121	0x81	Q	113	71	161	0x113	q
50	32	062	0x50	2	82	52	122	0x82	R	114	72	162	0x114	r
51	33	063	0x51	3	83	53	123	0x83	S	115	73	163	0x115	s
52	34	064	0x52	4	84	54	124	0x84	T	116	74	164	0x116	t
53	35	065	0x53	5	85	55	125	0x85	U	117	75	165	0x117	u
54	36	066	0x54	6	86	56	126	0x86	V	118	76	166	0x118	v
55	37	067	0x55	7	87	57	127	0x87	W	119	77	167	0x119	w
56	38	070	0x56	8	88	58	130	0x88	X	120	78	170	0x120	x
57	39	071	0x57	9	89	59	131	0x89	Y	121	79	171	0x121	y
58	3A	072	0x58	:	90	5A	132	0x90	Z	122	7A	172	0x122	z
59	3B	073	0x59	;	91	5B	133	0x91	[123	7B	173	0x123	{
60	3C	074	0x60	<	92	5C	134	0x92	\	124	7C	174	0x124	
61	3D	075	0x61	=	93	5D	135	0x93]	125	7D	175	0x125	}
62	3E	076	0x62	>	94	5E	136	0x94	^	126	7E	176	0x126	~
63	3F	077	0x63	?	95	5F	137	0x95	_	127	7F	177	0x127	DEL

Source: www.LookupTables.com

Dettaglio del Main

Lo studente dovrà implementare un codice assembly RISC-V strutturato con un main e relative funzioni, che dovranno essere definite al bisogno.

Il main dovrà elaborare l'unico input utente del programma, dichiarato come una variabile string *listInput* nel campo .data del codice RISC-V. Tale *listInput* dovrà contenere una serie di comandi separati da ~ (ASCII 126), dove ogni comando contiene una operazione ed eventualmente dei parametri. *listInput* non dovrà contenere più di 30 comandi. Nello specifico:

- ADD(char): crea un nuovo elemento della lista che contiene come informazione DATA=char, e viene aggiunto in coda alla lista esistente
- DEL(char): ricerca l'elemento con DATA=char esistente nella lista e, se esistente, lo elimina. Nel caso in cui più elementi con DATA=char siano presenti nella lista, li rimuove tutti.
- PRINT: stampa tutti i DATA degli elementi della lista, in ordine di apparizione, in modo ricorsivo
- SORT: ordinamento crescente della lista, da effettuarsi con procedura ricorsiva
- REV: inverte gli elementi della lista, da effettuare con ausilio dello stack

Lo studente dovrà predisporre un puntatore ad un'area di memoria che conterrà il primo elemento della lista. Questo puntatore indicherà il primo elemento della lista, e potrà essere modificato al bisogno durante il programma. E' prevista (ma non obbligatoria) la presenza di un puntatore alla coda della lista, ma non si prevedono altri puntatori di supporto.

Controllo Input

Si dovrà quindi inserire un controllo degli input e della formattazione dei singoli comandi. Questo deve essere definito dallo studente e dettagliato nella relazione.

- Nel caso delle operazioni ADD e DEL si suppone di avere uno ed uno solo carattere tra parentesi; nel caso in cui compaiano zero o più di un carattere tra parentesi, l'operazione si considera mal formattata e da scartare.
- I caratteri dei comandi devono essere consecutivi: sarà ammissibile il comando "SORT" ma non il "SO RT". Allo stesso modo, è ammissibile "ADD(d)" ma non "AD D(d)" o "ADD (d)"
- Due comandi anche ben formattati ma non separati da tilde sono da considerarsi come un unico comando mal formattato
- Spazi vicini alle ~ sono ammessi e devono essere tollerati dal programma.
- Il comando deve essere espresso con lettere maiuscole. "PRINT" è ammissibile, "print" non lo è.

Ogni comando mal-formattato dovrà essere scartato, e l'esecuzione del programma continuerà analizzando il successivo comando (se esistente).

Esempi di Input ed Esecuzione

listInput = "ADD(1) ~ ADD(a) ~ ADD(a) ~ ADD(B) ~ ADD(;) ~ ADD(9) ~ SORT ~ PRINT ~ DEL(b) ~ DEL(B) ~ PRI ~ REV ~ PRINT"

Comando Corrente	ADD(1)	ADD(a)	ADD(a)	ADD(B)	ADD(;)	ADD(9)	SORT	PRINT	DEL(b)	DEL(B)	PRI	REV	PRINT
Elementi in Lista	1	1a	1aa	1aaB	1aaB;	1aaB;9	;19aaB	;19aaB	;19aaB	;19aa	;19aa	aa91;	aa91;

listInput = "ADD(1) ~ ADD(a) ~ add(B) ~ ADD(B) ~ ADD ~ ADD(9) ~ PRINT ~ SORT(a) ~ PRINT ~ DEL(bb) ~ DEL(B) ~ PRINT ~ REV ~ PRINT"

Comando Corrente	ADD(1)	ADD(a)	add(B)	ADD(B)	ADD	ADD(9)	PRINT	SORT(a)	PRINT	DEL(bb)	DEL(B)	PRINT	REV	PRINT
Elementi in Lista	1	1a	1a	1aB	1aB	1aB9	1aB9	1aB9	1aB9	1aB9	1a9	1a9	9a1	9a1

Dettaglio delle Singole Operazioni

Sotto si fornisce il dettaglio delle singole funzioni da implementare per il progetto.

ADD - Inserimento di un Elemento

L'inserimento di un nuovo elemento nella lista (in coda) comporta principalmente quattro operazioni:

1. identificazione di una porzione di memoria da 5 byte che non si sovrappone con dati esistenti
2. la memorizzazione del nuovo elemento nella nuova area di memoria, con PAHEAD uguale a 0x00000000
3. l'aggiornamento del puntatore PAHEAD dell'elemento che era coda alla lista, che dovrà puntare all'elemento che stiamo inserendo
4. spostamento del puntatore alla coda (se previsto)

DEL - Rimozione di un Elemento

Questa operazione porta ad eliminare un dato elemento da una lista.

- Si deve identificare l'elemento (o gli elementi) della lista che corrisponde all'elemento da eliminare (se presente)
- Si deve ricavare l'indirizzo dell'elemento precedente rispetto a quello da rimuovere
- Modificare il PAHEAD dell'elemento precedente sovrascrivendolo con l'indirizzo all'elemento successivo rispetto a quello da rimuovere, o a 0x00000000 se un elemento diventa l'ultimo della lista.
- Aggiornamento dei puntatori alla testa ed alla coda (se presente) della lista

PRINT - Stampa della Lista

Questa funzionalità stampa il contenuto della lista, in ordine di apparizione dalla testa fino alla coda.

L'implementazione deve essere gestita in modo ricorsivo

```
print_list(list[0:n]) = { if len(list) > 0: print list[0]; print_list(list[1:n]); }
```

SORT - Ordinamento della Lista

Questa funzionalità ordina gli elementi della lista in base ad un dato algoritmo di ordinamento, da

effettuarsi preferibilmente (costituisce merito aggiuntivo) tramite un algoritmo di ordinamento ricorsivo (es. quicksort, mergesort).

REV - Inversione degli Elementi della Lista

Questa funzionalità inverte la lista esistente. Tale operazione può essere fatta sia modificando i puntatori degli elementi della lista (PAHEAD, e puntatori a testa e - se previsto - coda) sia cambiando l'informazione contenuta dagli elementi.

Note e Modalità di Consegna

Note

- Seguire fedelmente tutte le specifiche dell'esercizio (incluse quelle relative ai nomi delle variabili e al formato del loro contenuto).
- Rendere il codice **modulare** utilizzando ove opportuno **chiamate a procedure e rispettando le convenzioni fra procedura chiamante/chiamata**. La modularità del codice ed il rispetto delle convenzioni saranno aspetti fondamentali per ottenere un'ottima valutazione del progetto. Si richiede in particolare di *implementare ogni operazione (ciascun algoritmo ADD-DEL-PRINT-SORT-REV) come una procedura separata da chiamare con jal*.
- Commentare il codice in modo significativo (è poco utile commentare *addi s3, s3, 1* con "sommo uno ad s3"....).

Modalità di Esame

- Per sostenere l'esame è necessario consegnare preventivamente il codice e una relazione PDF sul progetto assegnato. Il progetto deve essere svolto dagli studenti singolarmente.
- Il codice consegnato deve essere funzionante sul simulatore RIZES in una versione uguale o maggiore alla 2.2.6, usato durante le lezioni.
- La scadenza esatta della consegna verrà resa nota di volta in volta, in base alle date dell'appello.
- Discussione e valutazione: la discussione degli elaborati avverrà contestualmente all'esame orale e prevede anche domande su tutti gli argomenti di laboratorio trattati a lezione.

Struttura della Consegna

La consegna dovrà consistere di un unico archivio contenente 3 componenti. L'archivio dovrà essere caricato sul sito moodle del corso di appello in appello, e dovrà contenere:

- Un unico file contenente il **codice** assembly
- un **breve video** (max 3 minuti) dove si registra lo schermo del dispositivo che avete utilizzato per l'implementazione durante l'esecuzione del programma, commentandone il funzionamento in base a 2-3 combinazioni di input diverse
- la **relazione** in formato PDF, strutturata come segue.
 1. **Informazioni** su autori, indirizzo mail, matricola e data di consegna
 2. **Descrizione** della soluzione adottata, trattando principalmente i seguenti punti:
 - a. Descrizione ad alto livello di ciascuna funzione della lista, di altre eventuali procedure e del main, in linguaggio naturale, con flow-chart, in pseudo-linguaggio, etc
 - b. Uso dei registri e memoria (stack, piuttosto che memoria statica o dinamica)
 3. **Test** per fornire evidenze del funzionamento del programma, anche in presenza di input errati o mal-formattati. Devono comparire **almeno** i test che usano gli input descritti nella parte di "Esempio"
 4. La relazione dovrà avere una **lunghezza massima di 5 pagine** (eccetto disegni e immagini) con

testo normale, bordi normali