PROVA SCRITTA DI CALCOLATORI ELETTRONICI DEL 16/9/2014
(Tempo a disposizione: 3,5 ore)

ESERCIZIO 1 (Tutti):
Si realizzi una rete sequenziale sincrona R con un ingresso X ed una uscita Z. La rete si comporta come una sorta di *selettore di uscita* sequenziale. In tre istanti di clock consecutivi, la rete riceve l'input, ossia l'ingresso, obbligatoriamente un 1 e due segnali di controllo. Qualora l'ingresso non sia un 1, la rete lo ignora e attende un 1.
La rete fornisce in uscita un 1 in corrispondenza di uno dei tre istanti di clock successivi alla ricezione dell'input sulla base dei segnali di controllo. In particolare, i segnali di controllo possono assumere valore 00, 01 o 10 e la rete restituisce un 1 rispettivamente in corrispondenza del primo, del secondo o del terzo istante di clock successivo alla ricezione dell'input. La configurazione 11 come coppia di segnali di controllo non è ammessa, qualora si presentasse in input la rete riprenderebbe il proprio funzionamento dal principio.
Segue un possibile funzionamento di R:

t:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
X:	0	**1**	0	1	**1**	0	0	**1**	1	0	0	0	**1**	0	1	**1**	1	1	0	0	**1**	...
Z:	0	0	0	0	0	1	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	...

La rete riceve il primo ingresso valido (ossia pari a 1) all'istante t=1, successivamente (istanti t=2 e t=3) riceve i segnali di controllo, pari a 01. Pertanto, dovrà restituire un 1 in corrispondenza del secondo istante di clock successivo alla ricezione dell'input, (quindi successivo all'istante t=3), ossia all'istante t=5. Nel frattempo, in corrispondenza dell'istante t=4, la rete riceve un nuovo ingresso valido e riceve 00 come segnali di controllo. Pertanto, la rete restituirà un 1 in corrispondenza del primo istante di tempo successivo alla ricezione dell'input (quindi successivo all'istante t=6), ossia all'istante t=7. Nel frattempo, all'istante t=7, riceve un nuovo ingresso valido e 10 come segnali di controllo. Quindi, la rete restituirà 1 in corrispondenza dell'istante t=12; e così via.

ESERCIZIO 2 (DM270 – A. A. 2012/13 e 2013/14):
Estendere il set di istruzioni della macchina a stack con l'operazione **LLARRAY,** definita come segue. L'istruzione costruisce una lista doppiamente collegata (ossia ogni suo nodo punta sia al precedente sia al successivo) a partire dalle informazioni contenute nell'array.
In cima allo stack è memorizzato il puntatore ad un array M. L'array contiene un insieme di indirizzi di memoria a partire dai quali memorizzare i nodi della lista. Ogni nodo della lista è composto da tre campi: (i) il valore memorizzato nel nodo, pari all'OR tra l'indirizzo del nodo e la posizione (a partire da 0) che il nodo assume nella lista; (ii) l'indirizzo del nodo precedente e (iii) l'indirizzo del nodo successivo. Alla fine dell'istruzione, il puntatore al primo elemento della lista deve essere memorizzato in cima allo stack.
La figura sulla sinistra mostra lo stato della memoria al termine dell'esecuzione dell'istruzione. Si noti che ad. esempio all'indirizzo 4056 è memorizzato il numero 4057 perché l'OR tra l'indirizzo del nodo (ossia 4056) e la sua posizione nella lista (ossia 1) è pari a 4057.

SP
1054

1052	8231
1053	7501
:	:
3542	3542
3543	4057
3544	-1
:	:
4056	4057
4057	7501
4058	3542
:	:
7501	7501
7502	-1
7503	4056
:	:
8228	3542
8229	4056
8230	7501
8231	2

ESERCIZIO 2 (DM270 – A. A. precedenti):
Estendere il set di istruzioni della macchina a registri con l'operazione **SUM@# Ri, Rj, Rk,** definita come segue.
A partire dalle locazioni i cui indirizzi sono memorizzati in R_i ed R_j, sono memorizzati due array, V_i e V_j, di 32 elementi. L'istruzione memorizza nel vettore V_k, posto in memoria a partire dalla locazione il cui indirizzo è memorizzato in R_k, la somma degli elementi dei vettori V_i e V_j utilizzando gli elementi di V_i come indirizzi indiretti e gli elementi di V_j come operandi immediati. Quindi, ogni elemento di V_i rappresenta un indirizzo di memoria nel quale è memorizzato l'indirizzo dell'elemento da sommare, mentre ogni elemento di V_j rappresenta direttamente l'intero da sommare. In formula: $V_k(i) = M[M[V_i(i)]] + V_j(i)$.
Si consideri il seguente esempio in cui Ri=947, Rj=1234 e Rk=3856 ed i vettori sono, per semplicità, di dimensione 4 anziché 32.

947	948	949	950	..	1234	1235	1236	1237	..	2431	..	3215	..	3856	3857	3858	3859	..	4023	:	8521	..	947
3215	2431	8521	4023	..	2	7	1	4	..	5	..	3	..	5	12	9	5	..	1	:	8	..	3215

ESERCIZIO 2 (DM509):
Estendere il set di istruzioni della macchina a stack con l'operazione **@SUM X,** definita come segue.
A partire dalla locazione X sono memorizzati due interi. L'intero memorizzato nella locazione X rappresenta l'indirizzo di memoria a partire dal quale è memorizzato un array la cui dimensione è specificata nella locazione X+1.
L'istruzione effettua la somma degli elementi multipli di 4 del vettore e memorizza tale valore nell'accumulatore.
Si consideri il seguente esempio in cui X=947. Al termine dell'esecuzione dell'istruzione @SUM X, l'accumulatore conterrà 48.

	1234	4		5	32	16	44	
	947	948		1234	1235	1236	1237	

ESERCIZIO 3 (DM 270 – 9CFU/6CFU):

Scrivere una procedura assembly che riceve un vettore di double word V, un vettore di word W e un vettore di word Z, tutti della stessa lunghezza.

Il primo vettore contiene indirizzi di memoria, in ciascuno dei quali è memorizzato un intero a 16 bit, mentre il secondo vettore contiene degli interi. La procedura memorizza in ogni elemento Z[i] la somma tra l'intero W[i] e l'intero memorizzato nell'indirizzo contenuto in V[i].

Scrivere inoltre il programma principale che invochi opportunamente la procedura descritta.