Nome, cognome, matricola .....

Calcolatori Elettronici (12AGA) – esame del 24.9.2016

Domande a risposta chiusa (è necessario rispondere correttamente ad almeno 6 domande).
Non è possibile consultare alcun tipo di materiale. Tempo: 15 minuti.

1	Quanti full-adder sono necessari per implementare un sommatore sequenziale in grado di sommare 2 numeri su 16 bit?			
2	Quanti transistor sono necessari per implementare una cella da 1 bit in una DRAM?	1	A	
		2	B	
		4	C	
		16	D	
3	Che cosa contiene il μPC in un'unità di controllo microprogrammata?	I segnali di controllo uscenti dalla memoria di microcodice	A	
		L'indirizzo della microistruzione corrente	B	
		L'indirizzo dell'istruzione corrente	C	
		Il codice operativo dell'istruzione corrente	D	
4	Quale dei seguenti dispositivi può diventare master di un bus?	Interfaccia di periferico	A	
		Memoria	B	
		Interrupt controller	C	
		DMA controller	D	
5	Quanti bit compongono l'Address Bus del processore 8086?	8	A	
		16	B	
		20	C	
		32	D	
6	Si consideri una cache con le seguenti caratteristiche • 128 linee da 16 byte • direct mapping • write-through. Assumendo che gli indirizzi emessi dal processore siano su 32 bit, in quale linea è memorizzata la parola con indirizzo esadecimale 0054 A40B?			
7	Quale tra le seguenti istruzioni x86 richiede il minor numero di colpi di clock per essere eseguita?	MOV AX, [BX]	A	
		ADD AX, 15	B	
		SUB [SI], 5	C	
		MUL CX	D	
8	Dove è memorizzata la Interrupt Vector Table in un sistema general-purpose basato su 8086?	Nella cache	A	
		In una ROM	B	
		Nella RAM	C	
		Nel TLB	D	
9	Si scriva un frammento di codice che calcola il valore del prodotto tra una variabile con segno in AX e un'altra (sempre con segno) in DX, scrivendo il risultato in memoria nella variabile VAR su 32 bit.			

Risposte corrette

1	2	3	4	5	6	7	8	9
1	A	B	D	C	64	B	C	

Domanda 9 (esempio di soluzione)

```
IMUL DX
MOV VAR, AX
MOV VAR+2, DX
```

Nome, cognome, matricola ….....……..…….....

Domande a risposta aperta (sino a 5 punti per ogni domanda) – Non è possibile consultare alcun materiale - Tempo: 40 minuti.

10	Si disegni l'architettura di un contatore sincrono e quella di un contatore asincrono, illustrando vantaggi e svantaggi di ciascuno.
11	Si descrivano le caratteristiche dei processori RISC.

12	Si elenchino nell'ordine le microistruzioni eseguite da un processore durante il fetch e l'esecuzione dell'istruzione SUB [R1], 15, che sottrae il valore 15 al contenuto della cella di memoria il cui indirizzo è contenuto nel registro R1.
13	Si descriva il funzionamento di una memoria che utilizza il meccanismo del codice di parità, disegnando l'architettura dell'hardware necessario e spiegando quali errori sono rilevati e quali corretti.

Nome, cognome, matricola ….....……..…….....

Esercizio di programmazione

sino a 12 punti – è possibile consultare qualunque materiale cartaceo - tempo: 60 minuti

Si scriva in linguaggio Assembly 8086 una procedura **riempiZaino** che trovi una soluzione *euristica* al problema dello zaino.
Sia dato uno zaino capace di sopportare un peso *W*. Sono disponibili *N* oggetti, ciascuno caratterizzato da un peso w_i e da un valore c_i. Si vuole individuare quali oggetti inserire nello zaino al fine di massimizzare il valore totale con il vincolo del peso complessivo, inferiore a *W*.

La procedura riceve tramite stack:

- l'offset del vettore *valore*, che specifica il valore (strettamente positivo) di ciascun oggetto
- l'offset del vettore *peso*, che specifica il peso (strettamente positivo) di ciascun oggetto
- l'offset del vettore *stato*, che indica se l'oggetto è stato già esaminato e l'eventuale decisione presa. Gli elementi di questo vettore possono assumere 3 valori:
  - 0: l'oggetto non è stato ancora esaminato
  - 1: l'oggetto è stato inserito nello zaino
  - -1: l'oggetto non può essere inserito nello zaino perché supera la capacità residua.
- il numero *contenuto*, espresso su un byte, che indica il peso degli oggetti attualmente presenti nello zaino.

Tutti i vettori sono composti da *N* elementi di tipo byte (*N* è dichiarata come costante, così come *W*). Non è ammesso l'uso di altre variabili.
La procedura **riempiZaino** deve:

1) trovare l'oggetto con il massimo *costo unitario* (definito come c_i / w_i) fra quelli il cui stato è 0. Per il calcolo del costo unitario, si consideri solo la parte intera del rapporto c_i / w_i. In caso di più oggetti con lo stesso costo unitario massimo, se ne prenda uno a scelta.
2) controllare se l'oggetto individuato sta nello zaino (ossia, il peso dell'oggetto sommato al peso degli oggetti già messi nello zaino è inferiore a W).
   a. Se l'oggetto sta nello zaino, la procedura aggiorna il peso degli oggetti attualmente presenti nello zaino (modificando il valore di *contenuto* nello stack) e pone a 1 lo stato dell'oggetto.
   b. Se l'oggetto non sta nello zaino, lo stato dell'oggetto è posto uguale a -1.

Di seguito un esempio di programma chiamante:

```
N EQU 9      ; numero di oggetti
W EQU 27     ; capacità dello zaino
.MODEL small
.STACK
.DATA
valore DB 200, 225, 250, 150, 125, 125, 130, 120, 75
peso   DB 10,  12,  15,  9,   7,   6,   6,   6,   4
stato DB N DUP (0)

.CODE
.STARTUP
PUSH OFFSET valore
PUSH OFFSET peso
PUSH OFFSET stato
MOV AX, 0
PUSH AX     ; valore iniziale di contenuto
MOV CX, N
ciclo:
CALL riempiZaino
LOOP ciclo
ADD SP, 8
.EXIT
```

Alla prima chiamata, l'oggetto considerato è quello in posizione 6 (con costo unitario massimo di 21) e l'oggetto viene inserito nello zaino, aggiornando il valore di *contenuto* a 6.
Al termine del programma, con i valori nell'esempio, il vettore *stato* è: 1, -1, -1, -1, -1, 1, 1, -1, 1. Gli elementi inseriti nello zaino, così come indicato nel vettore *stato*, hanno peso complessivo 26 e valore 455.