/* FUNZIONE esegui */

2 vettori con elementi interi a 16 bit

10 elementi ciascuno

esegui(addrvet1, addrvet2, numelem)

addrvet1: scostamento, rispetto all'inizio dello stack, dell'inizio del primo vettore
addrvet2: scostamento, rispetto all'inizio dello stack, dell'inizio del secondo vettore
numelem: numero di elementi interi a 16bit di cui e' composto ciascun vettore

esegui: restituisce un risultato intero mettendolo nel registro AX

/* FUNZIONALITA' DA IMPLEMENTARE IN esegui */

la funzione esegui prima chiama usa1 e successivamente usa2

usa1(addrvet1, numelem)

usa2(addrvet2, numelem)

esegui: somma dei risultati restituiti dalle due funzioni

/* FUNZIONALITA' DA IMPLEMENTARE IN usa1 e usa2 */

usa1: somma del quadrato di ciascun elemento del vettore che abbia indice pari (0, 2, 4, ...)
esempio. se il vettore e' cosi' composto (3,2,1,5,4,9,8,10,7,6)
il risultato deve essere 3*3 + 1*1 + 4*4 + 8*8 + 7*7 = 9 + 1 + 16 + 64 + 49 = 139

usa2: doppio del prodotto degli elementi del vettore che hanno indice compreso tra 0 e 3 (estremi compresi)
esempio. se il vettore e' cosi' composto (3,2,1,5,4,9,8,10,7,6)
il risultato deve essere 2*(3 + 2 + 1 + 5) = 2*11 = 22

usa1, usa2: ciascuna restituisce un intero a 16 bit, mettendolo in uno spazio dello stack creato dal chiamante nel record di attivazione costruito per la funzione stessa

/* VARIOUS */

vet1: (1, 3, 4, 7, 5, 8, 6, 4, 6, 2)

index: [0] [1] [2] [3] [4] [5] [6] [7] [8] [9]

usa1 = 1*1 + 4*4 + 5*5 + 6*6 + 6*6 =
= 1 + 16 + 25 + 36 + 36 =
= 114

vet2: (3, 11, 9, 7, 8, 0, 4, 2, 3, 1)

index: [0] [1] [2] [3] [4] [5] [6] [7] [8] [9]

usa2 = 2*
(3 + 11 + 9 + 7) = 2 * 30 = 60

esegui = 114 + 60 = 174

/----------/

--------usa1--------

usa1(addrvet1, numelem)

La prima istruzione della subroutine è PUSH del puntatore base BP, che servirà a indirizzare gli argomenti e le variabili locali della subroutine. Successivamente viene copiato il puntatore allo stack di modo che il nuovo valore del puntatore base punti al suo vecchio valore.
A questo punto vengono caricati nei registri gli argomenti. In SI l'offset, rispetto all'inizio del segmento di stack, dell'inizio del primo vettore e in CX il numero di elementi interi di quel vettore. Visto che ci interessa il numero degli elementi di indice pari viene effettuata l'istruzione SHR CX,1 che fa scorrere il valore di CX di un bit a destra (è come dividere per 2 ma si utilizza una sola istruzione e il risultato è contenuto nel registro stesso) così ora CX contiene il numero di elementi di indice pari del vettore.
A questo punto inizia il ciclo col caricare in BP il valore di SI che serve per indirizzare gli elementi del vettore.Le tre istruzioni successive riguardano il calcolo del quadrato dell'elemento del vettore. In AX viene caricato il valore dell'elemento puntato da BP utilizzato come registro indiretto e che quindi utilizza il segmento stack. Viene utilizzato DI per moltiplicare il valore per se stesso (quadrato) utilizzando l'istruzione MUL e quindi il registro implicito AX dove sarà contenuto il risultato.
L'istruzione successiva sposta il puntatore base in cima allo stack. Questo serve per riuscire ad indirizzare facilmente lo spazio dello stack che la funzione chiamante ha riservato per restituire il risultato. Viene aggiunto a questo spazio il valore corrente di AX.
Come ultima istruzione del ciclo viene incrementato il valore di SI di 4 perché essendo gli elementi del vettore valori interi a 16bit contenuti in posizioni dello stack (orientato alla parola e quindi con indirizzi di solito sempre pari) per accedere al prossimo elemento di indice pari bisogna incrementare di 4 e non di 2 che servirebbe per spostarsi solo all'elemento successivo.
Il ciclo viene eseguito per CX volte. Finito il ciclo viene ripristinato BP e viene passato il controllo al chiamante (esegui).

--------usa2--------

usa2(addrvet2, numelem)

Nelle prime 4 righe come di consueto viene impilato il valore di BP, fatto puntare al suo vecchio valore e successivamente caricati nei registri gli argomenti passati alla subroutine. In CX il numero degli elementi del vettore e in SI l'offset rispetto all'inizio dello stack dell'inizio del secondo vettore. La funzione deve sommare gli elementi del vettore di indice compreso tra 0 e 3 (estremi compresi), quindi i primi 4 elementi.
CX viene impostato a 4, AX viene inizializzato a 0 e DI a 2 per effettuare il prodotto.

Il ciclo effettua semplicemente la somma di questi elementi utilizzando (BP) e incrementando SI per indirizzarli.
Alla fine del loop il valore che risiede in AX (cioè il risultato della somma deglie elementi) viene moltiplicato per DI (2) e così abbiamo in AX il risultato finale che viene poi caricato nello spazio dello stack creato dal chiamante.
Viene effettuata la pop del puntatore base e poi si passa il controllo al programma chiamante.

--------esegui--------

esegui(addrvet1, addrvet2, numelem)

La funzione esegui si occupa di chiamare le due funzioni usa1 e usa2 e di restituire la somma dei risultati restituiti da esse.
Viene creato lo spazio per due word decrementando SP di 4. Le due word conterranno i rispettivi risultati delle due funzioni chiamate.
Per chiamare le due funzioni gli argomenti delle due subroutine vengono impilati in ordine inverso per rispettare la convenzione vigente per le chiamate de linguaggio C. Così facendo possono essere anche invocate all'interno di un programma C.
Ogni volta che le due subroutine ripassano il controllo ad essa, la procedura esegui ripulisce lo stack dai parametri passati ad esse.
Finite le due chiamate i due risultati vengono sommati in AX. Viene ripulito lo stack dai parametri passati, dai due risultati e viene effettuata la POP di BP e passato il controllo al programma principale.