RELAZIONE PROGETTO ASSEMBLY

Il nostro progetto è un codice scritto a basso livello che date due matrici in input da tastiera di dimensione qualunque (NxM) fino a un limite di 7x7, la dimensione massima può essere banalmente estesa, ne calcola il loro prodotto, la loro trasposta e in caso le matrici di input e/o di output siano 3x3 o 2x2 ne calcola anche il determinante. I numeri che si possono inserire da tastiera vanno da 0 a FF.

Un esempio di un possibile input è l i $2332010203040B0CA10102030405

Le prime due cifre sono le righe e colone di prima e seconda matrice, il riempimento delle stesse avviene per righe con numeri nel formato esadecimale.

Spieghiamo il codice

Iniziamo subito inizializzando lo stack pointer a un indirizzo molto alto per evitare che lo stack sia invaso dal codice stesso. Successivamente c'è una chiamata a una funzione TAKEMATRIXDIM, questa funzione ha lo scopo di prendere da input da tastiera le righe e le colonne che l'utente vuole inserire, qui di seguito illustreremo la struttura di questa funzione.

TAKEMATRIXDIM:

INB R1 0001 **questo comando mette in R1 la prima lettera messa da input col comando l i $ il punto è che viene preso l'ASCII del numero**

LDWA R0 ASCII **ho una variabile nel codice che ho chiamato ASCII ed è quel numero che sottratto all'ASCII dell'input mi da il valore esatto del numero che è stato inserito**

SUB R0 R1 **diciamo che togliamo l'ascii del numero per mantenerci il numero puro**

STWA R1 RIGHE1 **Conservo quindi R1 in una variabile word precedentemente impostata chiamata RIGHE1 la quale rappresenta il numero di righe della prima matrice**

INB R1 0001 **il ragionamento fatto per le altre variabili è identico**

LDWA R0 ASCII

SUB R0 R1

STWA R1 COLONNE1

INB R1 0001

LDWA R0 ASCII

SUB R0 R1

STWA R1 RIGHE2

INB R1 0001

LDWA R0 ASCII

SUB R0 R1

STWA R1 COLONNE2

RET **ritorniamo alla riga dopo la call**

A questo punto l'istruzione che verrà eseguita sarà un'altra chiamata a una funzione, ovvero una chiamata alla funzione CALCULATEDIM. Questa funzione ha lo scopo di calcolare il prodotto righe per colonne di

prima, seconda e terza (matrice risultante). Questo ci serve poiché le matrici in linguaggio a basso livello sono non altro che vettori per cui mi è utile sapere la sua dimensione.

CALCULATEDIM:

LDWA R1 RIGHE1 **Carico in R1 il valore presente nella variabile RIGHE1**

LDWA R2 COLONNE1 **Carico in R2 il valore presente nella variabile COLONNE1**

CALL MULTI **faccio una chiamata alla funzione multi la quale fa il prodotto R1*R2 e lo conserva in R10**

STWA R10 DIM1 **Conservo R10 in DIM1 ovvero il prodotto righe per colonne della prima matrice è conservato nella variabile word DIM1**

LDWA R1 RIGHE2 **Il ragionamento è identico per tutti questi altri casi**

LDWA R2 COLONNE2

CALL MULTI

STWA R10 DIM2

LDWA R1 RIGHE1

LDWA R2 COLONNE2

CALL MULTI

STWA R10 DIM3

RET

Ora dopo il RET la prossima istruzione sarà la chiamata alla funzione MATRIXFILL la quale ci ha fatto sbattere un po' la testa. Questa funzione ha lo scopo di analizzare tutti gli input, formattarli e salvarli nelle apposite posizioni delle matrici.

MATRIXFILL:

LDWI R0 MAT1 **Carico in R0 l'indirizzo dell'etichetta MAT1 che coincide all'indirizzo zero della matrice**

LDWA R3 DIM1 **Carico in R3 il valore contenuto nella variabile DIM1**

LDWA R11 ASCII **Carico in R11 il valore contenuto nella variabile ASCII**

LDWA R1 HEX **Carico in R1 il valore contenuto nella variabile Hex, essa contiene (in decimale) 16, mi serve per formattare i numeri, ad esempio 10 in esadecimale è uguale a 1*16 + 0 = 16**

QUAA: LDWA R13 UNO **Carico in R13 il valore contenuto nella variabile UNO, la qual contiene 1, questo registro è quello che fa si che la prima cifra ricevuta d input venga moltiplicata per 16 mentre la seconda cifra no, in questo modo formatto i numeri nel formato da me desiderato, in seguito capiremo perchè**

LDWI R12 0 **Rendo R12 uguale a zero**

LOOP_A: INB R2 0001 **Qui arrivano i numeri inseriti da tastiera predisposti ad essere gli elementi delle matrici, per cui qui dobbiamo formattarli, sapendo che ogni numero può essere un numero da 0 a F**

LDWI R4 HEXA **Faccio si che R4 punti all'indirizzo di HEXA il quale è un vettore che contiene delle word che rappresentano l'ASCII di 0A, 0B, 0C, 0D, 0E, 0F. Questo vettore mi serve per controllare se l'utente passa da tastiera un numero che non ha solo cifre decimale ma anche delle lettere caratteristiche dell'esadecimale**

LDWA R6 DIMHEXA **Ovviamente carico in R6 la dimensione del vettore HEXA, la quale è conservata nella variabile DIMHEXA.**

LDWI R7 0 **Setto R7 a zero**

Qui l'idea è scorrere il vettore hexa e vedere se da input c'è una lettera del tipo A,B,C,D,E,F. Tutto ciò è necessario poiché da input possiamo prendere solo l'ASCII delle cose che mettiamo e non il numero puro che quei caratteri rappresentano. Per i numeri da 0 a 9 è sufficiente fare una sottrazione, ma per le lettere che stiamo considerando, bisogna prima riconoscerle e non apriori sottrarre qualcosa. Questa naturalmente è la nostra idea, il nostro modo per risolvere questo problema, sicuramente ci sono modi migliori nel farlo.

CNTLOOP_A: LDWR R5 R4 **vado all'indirizzo puntato da R4 e metto il valore contenuto in quell'indirizzo di memoria in R5. (R4 punta al vettore HEXA)**

SUB R2 R5 **Controllo se R2 cioè l'input meno R5 fa zero cioè controllo se siano uguali o no. In caso siano uguali allora salterò all'etichetta CONVERT_A. Di seguito spiegheremo a cosa serve**

JMPZ CONVERT_A

INC R7 **Faccio doppio incremento a R7 e R4 poiché stiamo maneggiando delle word le quali in memoria occupano due celle**

INC R7

INC R4

INC R4

DEC R6 **Decremento R6 finchè non è zero, a quel punto vorrà dire che il vettore è stato completamente spazzolato e quindi saranno finiti i controlli per uanto riguarda caratteri del tipo A,B,C,D,E,F e quindi salto la conversione**

JMPNZ CNTLOOP_A

JMP LI_A

CONVERT_A: LDWI R6 HEXADEC **Qui si arriva solo se ci sono delle lettere del tipo A,B,C,D,E,F. Si fa puntare R6 all'indirizzo del vettore HEXADEC il quale contiene in word i valori numerici 0A,0B,0C,0D,0E,0F. In particolare questo vettore è ordinato come il vettore HEXA quindi siccome R7 contiene quel numero da aggiungere all'indirizzo iniziale di HEXA per arrivare all'elemento corrente, allora se aggiungo R7 all'indirizzo iniziale di HEXADEC arriverò esattamente al valore numerico corrispondente all'ASCII dell'input.**

ADD R6 R7 **Quindi faccio R6+R7 in R7**

LDWR R2 R7 **Quindi carico in R2 quello che c'è all'indirizzo puntato da R7**

ADD R11 R2 **Ora aggiungo a R2 R11 per necessità progettuali in quanto la prossima istruzione sarà per forza un fare R2-R11 e quindi siccome io voglio che arrivato qui R2 abbia già l'input formattato questo è quanto. Avremmo potuto fare un'etichetta che salta al rigo dopo la sub ma comunque per evitare mille etichette abbiamo optato per questa cosa.**

LI_A: SUB R11 R2 **Qui si arriva in ogni caso, o che il controllo vada a buon fine o che non lo faccio, in caso si arrivi con il controllo per le lettere negativo si toglie l'ASCII in caso il controllo vada a buon fine si toglie semplicemente una quantità che è stata aggiunta in precedenza**

DEC R13 **Ricordo che R13 vale uno quindi se lo decremento vale zero e quindi la prima volta il jmpnz fallisce, alla seconda iterazione varrà -1 e quindi la multi sarà saltata. La prima volta il codice chiamerà quindi multi la quale si troverà in R1 la vaiable HEX che contiene 16 in decimale e la moltiplicherà con R2 cioè l'input formattato che arriva da tastiera. Il risultato andrà in R10.**

JMPNZ LAA

CALL MULTI

ADD R10 R12 **Aggiungo a R12 R10, poiché R12 era uguale a zero allora in questo caso stiamo facendo una copia di R10, questo lo faccio perché fin qui è tutto una somma parziale, alla fine il primo elemento della matrice è la somma del primo input moltiplicato per 16 + il secondo input formattato**

JMP LOOP_A **Qui si salta all'etichetta LOOP_A cioè si rifà tutto questo solo che R13 varrà ad un certo punto -1 e farà saltare il codice all'etichetta qui in basso ovvero LAA**

LAA: ADD R2 R12 **Qui si aggiunge quindi a R12 che contiene la prima somma parziale R2 che contiene il secondo input formattato. A questo punto R12 conterrà esattamente il numero che l'utente ha inserito e può essere conservato.**

STWR R12 R0 **Conservo appunto R12 all'indirizzo puntato da R0, esso infatti punta al primo indirizzo della prima matrice.**

INC R0 **Doppio incremento per R0 perché la matrice uno è una matrice di word e quindi un'elemento successivo si troverà a due indirizzi di distanza**

INC R0

DEC R3 **Naturalmente tutto questo va fatto finchè R3 che contiene la dimensione della prima matrice non è zero. Quando sarà zero allora avremmo finito.**

JMPNZ QUAA

LDWI R0 MAT2 **Qui iniziamo a fare esattamente la stessa cosa spiegata pocanzi ma con la seconda matrice per cui non c'è bisogno di commentare.**

LDWA R3 DIM2

LDWA R1 HEX

LDWA R11 ASCII

QUAB: LDWA R13 UNO

LDWI R12 0

LOOP_B: INB R2 0001

LDWI R4 HEXA

LDWA R6 DIMHEXA

LDWI R7 0

CNTLOOP_B: LDWR R5 R4

SUB R2 R5

JMPZ CONVERT_B

INC R7

INC R7

INC R4

INC R4

DEC R6

JMPNZ CNTLOOP_B

JMP LI_B

```
CONVERT_B: LDWI R6 HEXADEC
           ADD R6 R7
           LDWR R2 R7
           ADD R11 R2
LI_B:      SUB R11 R2
           DEC R13
           JMPNZ LAB
           CALL MULTI
           ADD R10 R12
           JMP LOOP_B
LAB:       ADD R2 R12
           STWR R12 R0
           INC R0
           INC R0
           DEC R3
           JMPNZ QUAB
RET
```

Una volta finita questa funzione le nostre matrici sono riempite in memoria e pronte all'uso.

Prima di fare il prodotto però abbiamo bisogno di sapere se le due matrici sono compatibili, verificare questo è molto semplice infatti facciamo così:

```
LDWA R1 COLONNE1
LDWA R2 RIGHE2
SUB R1 R2
JMPZ OK
CALL ERRORLOOP
HLT
```

È molto semplice il ragionamento, affinchè le matrici siano compatibili il numero di colonne della prima deve essere uguale al numero righe dell'altra, per cui faccio una Sub e se ho zero allora salto all'etichetta OK dove poi si inizierà a fare il prodotto, altrimenti chiamerò la funzione ERRORLOOP la quale mi stamperà la scritta matrici non compatibili e il programma si fermerà. La funzione ERRORLOOP è un semplice spazzolamento di un vettore dove ogni elemento è una lettera e per ogni elemento faccio un OUTB.

A questo punto siamo all'etichetta OK, qui diciamo siamo nel cuore del codice dove faremo il prodotto tra le due matrici. L'idea qui è stata emulare ciò che facevamo in linguaggio C, ovvero cercare di costruire 3 for uno dentro l'altro. Specifichiamo che la variabile "i" equivale a RIGHE1-1 fino a 0, la variabile "j" equivale a COLONNE2-1 fino a 0, mentre "k" è la dimensione comune delle due matrici che va COLONNE1-1 a 0.

Questo perché la matrice risultate sarà del tipo RIGHE1*COLONNE2,

Quindi:

OK: LDWA R0 RIGHE1 **In R0 ci va il valore contenuto in RIGHE1**

LDWI R6 MAT3 **Faccio puntare R6 all'indirizzo iniziale di MAT3 ovvero la matrice risultante del prodotto**

LDWA R1 DIM3 **Metto in R1 il valore contenuto in DIM3**

DEC R1 **Lo decremento**

LSH R1 **Lo moltiplico per due perché in C facevamo andara ad esempio i da 0 a dim-1 ma poiché la matrice è in word allora la dimensione non sarà dim-1 ma (dim-1)*2**

ADD R1 R6 **Faccio puntare R6 all'ultimo elemento di MAT3 poiché in questo codice procederemo al contrario, andremo da dim-1 a 0 per dirla in C-mode**

DEC R0 **Decrementando R0 faremo in C-mode i-1**

FORI: LDWA R1 COLONNE2 **Quando arriviamo nel "for i " dobbiamo inizializzare j perché "for j" partirà da j-1**

naturalmente in R1 ci va il contenuto di COLONNE2.

DEC R1 **Decrementando R1 faremo in C-mode j-1**

FORJ: LDWA R2 COLONNE1 **Quando arrivo nel "for j" devo inizializzare k poiché "for k" partirà da k-1**

In R2 ci va il contenuto di COLONNE1.

DEC R2 **Decrementano quindi R2 faremo in C-mode k-1**

Qui nel FORK si fanno i conti, dovremmo fare l'operazione mat3[i][j]+=mat1[i][k]*mat2[k][j].

Una cosa degna di nota è che non dobbiamo calcolare l'offset di mat3 poiché essa viene riempita dall'ultimo elemento fino al primo. Il motivo per cui non ho bisogno di calcolare l'offset è perché nel FORK l'unica cosa a variare è k e non i e j per cui l'offset di mat3 rimane costante in tutto il FORK, poi una volta usciti dal FORK andremo a decrementare j e questo vuol dire semplicemente tornare indietro di un indirizzo (in realtà due perché sono word) per mat3 che è linearizzata, per cui l'offset di mat3 sarà sempre R6 e quando decremento j devo decrementare anche R6.

FORK: LDWI R10 MAT1 **Predispongo il tutto per calcolare l'offset di mat1 e mat2. Faccio puntare R10 al primo indirizzo di MAT1**

LDWI R11 MAT2 **Faccio puntare R11 al primo indirizzo di MAT2**

Calcolo prima l'offset di MAT1 la quale ha come numero di colonne COLONNE1, dobbiamo trovare quindi mat1[i][k].

LDWA R4 COLONNE1 **Metto in R4 il contenuto di COLONNE1, lo faccio perché l'offset si calcola nel seguente modo, OFFSET=INDIRIZZOINIZIALEMATRICE+i*COLONNETOTALIMATRICE+j nel nostro caso però poiché la matrice è in word allora l'offset sarà INDIRIZZOINIZIALEMATRICE+2*i*COLONNETOTALIMATRICE+2*j**

Nel caso specifico quindi l'offset sarà indirizzoinizialemat+2*i*colonnetotali+2*k

ADD R2 R10 **In queste due righe faccio l'Indirizzo iniziale della matrice +k +k**

ADD R2 R10

LDWI R5 0 **Qui di seguito faccio invece i*colonnetotalimatrice, il risultato andrà in R5**

LOOPX2: ADD R0 R5

DEC R4

JMPNZ LOOPX2

LSH R5 **Moltiplico per due R5 che contiene il prodotto i*colonnematrice, lo faccio per quanto specificato prima**

ADD R5 R10 **Concludo il calcolo dell'offset, qui infatti R10 conterrà esattamente l'indirizzo che punta all'elemento mat1[i][k]**

LDWA R8 COLONNE2 **Ora invece calcolo l'offset di MAT2 quindi per arrivare a MAT2[k][j] devo fare**

Indirizzoinizialemat2+2*k*colonnetotalimatrice+2*j

ADD R1 R11 **Qui faccio indirizzo inizialematrice + j + j**

ADD R1 R11

LDWI R5 0

LOOPX3: ADD R2 R5 **Faccio colonnetotalimatrice*k e il risultato andrà in R5**

DEC R8

JMPNZ LOOPX3

LSH R5 **Moltiplico per 2 R5**

ADD R5 R11 **Conclude il calcolo dell'offset.**

A questo punto R10 punterà ad Mat1[i][k] mentre R11 punterà a Mat2[k][j], quindi devo solo prendere quello presente a quegli indirizzi, moltiplicarli fra loro e il risultato sarà aggiunto a ciò che era già presente in mat3[i][j]

LDWR R8 R10 **R8 conterrà quindi l'elemento in mat1[i][k]**

LDWR R9 R11 **R9 conterrà l'elemento contenuto in mat2[k][j]**

LDWI R7 0 **Li moltiplico (R8*R9) il risultato andrà in R7**

LOOPX4: ADD R8 R7

DEC R9

JMPNZ LOOPX4

LDWR R15 R6 **Prendo l'elemento contenuto in mat3[i][j] e lo metto in R15**

ADD R15 R7 **Qui in pratica faccio mat3[i][j]+=mat1[i][k]*mat2[k][j] il risultato andrà in R7**

STWR R7 R6 **Conserverò quindi questo risultato in mat3[i][j], ricordo ancora che R6 è sempre l'offset corretto di mat3**

DEC R2 **K--**

JMPNN FORK **Finchè K non è negativo salta, come il for quindi, for(k=dim-1, k>=0, k--) {istruzioni}**

DEC R6 **Decremento di 2 l'offset di mat3 poiché la matrice è in word quindi l'elemento successivo sarà 2 indirizzi dietro**

DEC R6

DEC R1 **j--**

JMPNN FORJ **Finchè J non è negativo salta, C-mode.**

DEC R0 **I--**

JMPNN FORI **Finchè i non è negatico salta, C-mode.**

A questo punto dopo l'esaurimento di tutti i cicli Mat3 sarà riempita correttamente con il prodotto tra le due matrici generiche riempite da input il gioco è fatto, a questo punto facciamo stampare gli elementi di mat3, cosa abbastanza facile:

```
MAT3LOOP: LDWR R1 R0
        OUTW R1 0000
        INC R0
        INC R0
        DEC R2
        JMPNZ MAT3LOOP
```

A questo punto ci chiedevamo se questo fosse abbastanza, la risposta è stata: No. Allora siccome volevamo ancora lavorare con le matrici abbiamo optato al calcolo del determinante. Ci siamo limitati a matrici 2x2 o 3x3 perché calcolare il determinante per matrici di ordine superiore sarebbe stato un qualcosa di molto laborioso e forse non fattibile. Tornando a noi abbiamo costruito un funzione che dati in input le righe, colonne e indirizzo iniziale della matrice se la matrice è 3x3 o 2x2 allora ne fa il determinante. Le righe vanno passate in R1, le colonne in R2 e l'indirizzo iniziale in R0. Spieghiamo la funzione:

DETERMINANTE:

```
        LDWI R6 06 Rendo R6=6
        LDWI R5 03 R5=3
        LDWI R4 04 R4=4
        LDWI R10 0 R10=0
        LDWI R12 0 R12=0
        ADD R1 R10 In R10 copierò R1
        ADD R2 R12 In R12 copierò R2
        SUB R5 R1 Controllo se le righe sono 3
        JMPNZ DET2X2 se non sono 3 allora vado a vedere se sono 2
        SUB R5 R2 Vedo se le colonne sono 3
        JMPNZ DET2X2 Se non lo sono vedo se sono 2
        JMP DET3X3 Se arrivo qui vuol dire che righe e le colonne della matrice messa da input sono 3 per cui la
        matrice è 3x3 e ne posso calcolare il determinante
DET2X2: LDWI R5 02 Qui si arriva se fallisce il controllo per le 3x3; R5=2
        SUB R5 R10 Controllo che le righe siano 2
        JMPNZ FINE1 Se non lo sono non fare il determinante ed esce dalla funzione
        SUB R5 R12 Controllo che le colonne siano 2
        JMPNZ FINE1 Se non lo sono non fare il determinante ed esce dalla funzione
        LDWR R1 R0 Se sono qui allora la matrice è 2x2, il suo determinante sarà mat[0][0]*mat[1][1]
```

-mat[0][1]*mat[1][0]. Non ho quindi bisogno di calcolare l'offset poiché mi basta semplicemente saltare. Qui mi prendo mat[0][0] e lo metto in R1

ADD R6 R0 **Qui invece saltando di 6 indirizzi (essendo in word sono 3) andrò da mat[0][0] a mat[1][1]**

LDWR R2 R0 **Qui prendo mat[1][1] e lo metto in R2**

CALL MULTI **Moltiplico R1*R2 (mat[0][0]*mat[1][1]) e il risultato andrà in R10**

LDWI R11 0

ADD R10 R11 **Copio R10 in R11**

SUB R5 R0 **Ora sottraendo a R0 R5 torno indietro di 2 indirizzi (di uno perché usiamo le word) per cui se prima R0 puntava a mat[1][1] ora punterà a mat[1][0]**

LDWR R1 R0 **Metto quindi il contenuto di mat[1][0] in R1**

SUB R5 R0 **Torno di nuovo indietro di due indirizzi così R0 punta a mat[0][1]**

LDWR R2 R0 **Metto il contenuto di Mat[0][1] in R2**

CALL MULTI **Faccio R1*R2 e il risultato va in R10**

SUB R10 R11 **Faccio il Determinate parziale di prima meno quello di adesso il quale è proprio il determinante reale della matrice**

OUTW R11 0000 **Lo stampo a video.**

FINE1: BR FINE **Fine della funzione.**

DET3X3: LDWI R1 MATDET **Se sono qui la matrice è 3x3. Faccio puntare R1 all'indirizzo iniziale della matrice MATDET una matrice precedentemente predisposta da me fatta ad hoc per memorizzare la matrice che arriva qui in modo tale da poter calcolare il determinante senza offset ma facendo dei semplici salti**

LDWI R14 04 **R14=4**

LDWI R10 02 **R10=2**

LOOP_C: LDWI R15 03 **R15=3**

DEC R14 **Decremento R14 il motivo sarà più chiaro in seguito**

JMPZ SUM1 **se R14 è zero allora salta a SUM1**

L'idea è di memorizzare la matrice 3x3 come una matrice 3x5 ove le 2 colonne aggiuntive sono la copia delle prime due.

LOOP_D: LDWR R2 R0 **R0 punta all'indirizzo iniziale della matrice, prendo ciò che c'è a quell'indirizzo e lo metto in R2.**

STWR R2 R1 **Conservo R2 all'indirizzo puntato da R1**

ADD R10 R0 **Doppio incremento a R0**

ADD R10 R1 **Doppio incremento a R1**

DEC R15 **Decremento R15 perché questo va fatto 3 volte ma alla terza volta R1 deve incrementarsi ma R0 deve tornare indietro di 3 indirizzi (quindi di 6 unità), dopo questa operazione la copia da una matrice all'altra deve avvenire due volte dopo di che R0 deve tornare al valore che aveva qui per poi essere incrementato**

JMPNZ LOOP_D

SUB R4 R0 **Decremento R0 di 4 unità**

SUB R10 R0 **Decremento ancora R0 di altre 2 unità, quindi R0 ora punterà di nuovo al primo elemento della riga corrente**

LDWR R2 R0 **Metto in R2 ciò che punta R0**

STWR R2 R1 C**onservo R2 all'indirizzo puntato da R1**

ADD R10 R0 **Incremento R0 di due unità**

ADD R10 R1 **Incremento R1 di due unità**

LDWR R2 R0 **Metto in R2 il valore puntato dal registro R0**

STWR R2 R1 **Conservo R2 all'indirizzo puntato da R1**

ADD R10 R1 **Incremento R1 di due unità**

ADD R4 R0 **Incremento R0 di 4 unità quindi a questo punto R0 è tornato al valore che aveva prima di entrare in questa parte di codice**

JMP LOOP_C **qui si salta a LOOP_C da notare che R14 sancisce il fatto che tutto ciò va fatto 3 volte**

Dopo 3 volte si arriva qui, a questo punto la matrice MATDET sarà 3x5 identica alla matrice 3x3 iniziale ma le colonne aggiuntive sono la copia delle prime due colonne. Il determinante ora può essere calclato senza offset ma solo con dei salti. Devo fare a1,1·a2,2·a3,3 + a1,2·a2,3·a3,1 + a1,3·a2,1·a3,2 e a questo sottrarre

a1,3·a2,2·a3,1 - a1,1·a2,3·a3,2 - a1,2·a2,1·a3,3. Per fare questi conti mi basta solo fare dei salti con la matrice costruita ad hoc.

SUM1: LDWI R6 0C **R6 = 12 (decimale)**

LDWI R15 0 **R15=0**

LDWI R3 03 **R3=3**

CALC1: LDWI R0 MATDET **Faccio puntare R0 all'indirizzo di MATDET**

ADD R15 R0 **A ogni ciclo quando torno qui R0 deve partire non dall'indirizzo inziale di MATDET ma da R0+R15 che tiene conto delle iterazioni, mi servirebbe una lavagna per spiegare perchè**

LDWR R1 R0 **Metto in R1 il valore che punta R0**

ADD R6 R0 **Incremento R0 di 12 quindi di 6 indirizzi word perché è lì che si trova l'altro elemento che mi serve**

LDWR R2 R0 **Prendo l'elemento puntato da R0 e lo metto in R2**

CALL MULTI **R1*R2 e il risultato va in R10**

LDWI R1 0 **R1=0**

ADD R10 R1 **Copio R10 in R1**

ADD R6 R0 **Aggiungo altre 12 unità a R0 per prendere l'altro elemento che mi interessa**

LDWR R2 R0 **Metto in R2 il valore puntato da R0**

CALL MULTI **Faccio R1*R2 e il risultato va in R10**

LDWA R11 SOMMAP **Metto in R11 il valore della variabile SOMMAP**

ADD R11 R10 **Sommo a R10 R11**

STWA R10 SOMMAP **Conserva R10 in SOMMAP**

INC R15 **Incremento R15 di due unità così alla prossima iterazione R0+R15 punterà al secondo elemento della matrice MATDET**

INC R15

DEC R3

JMPNZ CALC1 **Ovviamente tutto questo va fatto 3 volte, infatti R3 vale 3, dopo 3 iterazione varrà zero e si andrà avanti. Usciti da qui SOMMAP conserverà a1,1·a2,2·a3,3 + a1,2·a2,3·a3,1 + a1,3·a2,1·a3,2**

LDWI R15 0 **R15=0**

LDWI R6 03 **R6=3**

Ora il ragionamento che si fa per l'ultima parte del determinante è identico a quello precedente, ciò che cambia sono i salti, se prima incrementavo di 12 qui devo incrementare di 8 e partire del secondo elemento di MATDET.

LOOP_E:LDWI R0 MATDET **Faccio puntare R0 all'indirizzo MATDET**

ADD R4 R0 **Incremento di 4 unità R0 quindi ora punterà al secondo elemento di MATDET**

ADD R15 R0

LDWR R1 R0

ADD R4 R0

ADD R4 R0

LDWR R2 R0

CALL MULTI

LDWI R1 0

ADD R10 R1

ADD R4 R0

ADD R4 R0

LDWR R2 R0

CALL MULTI

LDWA R11 SOMMAN

ADD R11 R10

STWA R10 SOMMAN

INC R15

INC R15

DEC R6

JMPNZ LOOP_E

LDWA R10 SOMMAN

LDWA R11 SOMMAP

SUB R10 R11 **Alla fine faccio la differenza tra questi due determinanti parziali e la conservo in R11**

OUTW R11 0000 **Stampo a video il Determinante**

FINE: LDWI R10 0

STWA R10 SOMMAN **Faccio ritornare le variabili di Somma a 0 per non commettere errori in successive chiamate alla stessa funzione**

STWA R10 SOMMAP

RET

A questo punto dopo aver preso la mano con il linguaggio non potevamo che fare anche la trasposta delle matrici. Di base la trasposta di una matrice NxM è una matrice MxN tale che B[i][j]=A[j][i].

Per cui abbiamo costruito una funzione che dati in input Righe, Colonne, indirizzo iniziale della matrice passati rispettivamente nei registri R1,R2,R6 ne stampa la trasposta. Di seguito illustreremo e spiegheremo la funzione.

TRASPONI:

STWA R1 RIGHEF **Conservo R1 nella variabile da me predisposta RIGHEF**

STWA R2 COLONNEF **Conservo R2 nella variabile COLONNEF, esso mi tiene conto delle iterazioni da fare nel FORI2**

DEC R1

DEC R2

LDWI R9 0

ADD R6 R9 **Copio R6 in R9**

STWA R9 NULL **Conservo R9 nella variabile NULL**

Qui l'idea è di emulare quello che si fa in C, ovvero 2 for annidati con l'istruzione B[i][j]=A[j][i] con i che va da 0 a dim-1 e j che va da 0 a dim-1. Nel nostro caso la i va da 0 a COLONNE-1 e j va da 0 a RIGHE-1

LDWI R5 0 **R5 prende il posto della i, C-mode**

FORI2: LDWI R10 0 **R10 prende il posto di j, C-mode**

LDWA R1 RIGHEF **R1 mi tiene conto delle iterazioni da fare nel FORJ2**

DEC R1

Ora per ogni iterazione devo avere un offset, il quale sarà calcolato come fatto in precedenza.

Per MATTRAS l'offset sarà indirizzoinizialematrice+2*i*COLONNETOT+2*j

Per la matrice in entrata invece sarà indirizzoinizialematrice+2*j*COLONNETOT+2*i

FORJ2: LDWI R0 MATTRAS

LDWA R6 NULL

ADD R10 R0 **Con questa doppia add faccio indirizzoiniziale della matrice +2*j poiché R10 rappresentano le j e R0 rappresenta indirizzoinizialematrice**

ADD R10 R0

LDWA R3 RIGHEF **Le colonne totali di MATTRAS sono appunto le righe della matrice di entrata**

LDWI R12 0

Faccio R5*R3 e il risultato va in R12, faccio in pratica COLONNETOT*i poiché R5 sono le mie i C-mode

LOOP_F: ADD R5 R12

DEC R3

JMPNZ LOOP_F

LSH R12 **Moltiplico per due R12**

ADD R12 R0 **A questo punto concludo il calcolo dell'offset e R0 punterà esattamente all'indirizzo i-j esimo di MATTRAS**

ADD R5 R6 **Qui ora con analoghe considerazioni calcolo l'offset della matrice di entrata**

ADD R5 R6

LDWA R3 COLONNEF

LDWI R12 0

LOOP_F2: ADD R10 R12

DEC R3

JMPNZ LOOP_F2

LSH R12

ADD R12 R6 **Qui si conclude il calcolo del secondo offset e R6 punterà all'elemento mat[j][i]**

LDWR R8 R6 **A questo punto devo prendere l'elemento puntato da R6 cioè devo prendere l'elemento j-i esimo della matrice di entrata e collocarlo all'elemento i-j esimo di MATTRAS**

STWR R8 R0 **Conservo appunto R8 all'indirizzo puntato da R0 cioè al posto i-j esimo di MATTRAS**

OUTW R8 0000 **Lo stampo subito senza poi andare a fare un loop esterno**

INC R10 **j++ C-mode**

DEC R1 **finchè R1 non è negativo salta a FORJ2, poiché R1 tiene conto della dimensione delle J**

JMPNN FORJ2

INC R5 **i++**

DEC R2 **finchè R2 non è negativo salta a FORI2, poiché R2 tiene conto della dimensione delle I**

JMPNN FORI2

RET

Questo è quanto. Questa è la fine del nostro progetto l'unica cosa che vorrei specificare è che quelle scritte tipo

LDWI R0 0

OUTB R0 0000

Sono messe per l'idea di mettere uno spazio tra un'operazione e l'altra, per esempio come output per matrici compatibili avremo:

Prodottomatriciale

00

Determinante prima matrice se è 3x3 o 2x2 altrimenti 00

Determinante seconda matrice se è 3x3 o 2x2 altrimenti 00

Determinante terza matrice se è 3x3 o 2x2 altrimenti 00

Trasposta prima matrice

Trasposta seconda matrice

Trasposta terza matrice

Ad ogni modo questo è tutto.