Il Livello di Microarchitettura (i)

A cura di:
Giuseppe Pozzi Donatella Sciuto

DEI, PoliMI, Milano

giuseppe.pozzi,donatella.sciuto@polimi.it

- versione dell'11 aprile 2003 -

Il livello di microarchitettura

Il livello di microarchitettura

Bibliografia:

- Tanenbaum A. S., Goodman J. R, "Architettura dei computer - Un approccio strutturato", Prentice Hall, 2000.
- Hamacher V. C., Vranesic Z. G., Zaky S. G., "Introduzione all'architettura dei calcolatori", McGraw-Hill, 1997 (capitolo 3).

Microarchitettura

- Definizione:
  - la microarchitettura si appoggia sul livello logico ed interpreta le istruzioni definite all'interno dell'insieme delle istruzioni eseguibili da quel processore (detto Instruction Set Architecture, ISA).
- Non esistono microarchitetture "standard", ma famiglie di microarchitetture (ad es. Intel, Motorola …), cui corrispondono i relativi instruction set.

Esempio di microarchitettura

- Durante il ciclo di lezioni sulla microarchitettura, come esempio di ISA analizzeremo un processore su cui funziona IJVM:
  - Integer - considera solo istruzioni intere, cioè per ora consideriamo solo istruzioni su operandi interi. E' un sottoinsieme dell'ISA;
  - JVM - Java Virtual Machine.
- La macchina con IJVM è un processore didattico.

La microarchitettura

- La microarchitettura prevede:
  - microprogramma presente nella memoria a sola lettura (ROM);
  - lettura delle singole istruzioni, che compongono il programma da eseguire, dalla memoria centrale (RAM) - fase detta di fetch;
  - decodifica (riconoscimento) della singola istruzione, specificata tramite il suo codice operativo (op-code);
  - esecuzione della singola istruzione.

Componenti della microarchitettura

- Lo stato del processore è definito da un insieme di variabili, che possono essere modificate tramite opportune funzioni.
- Ogni variabile è memorizzata in un registro, composto da bistabili montati sullo stesso chip del processore.
- Una opportuna variabile tiene traccia della prossima istruzione da eseguire (PC - program counter).

Il formato delle istruzioni

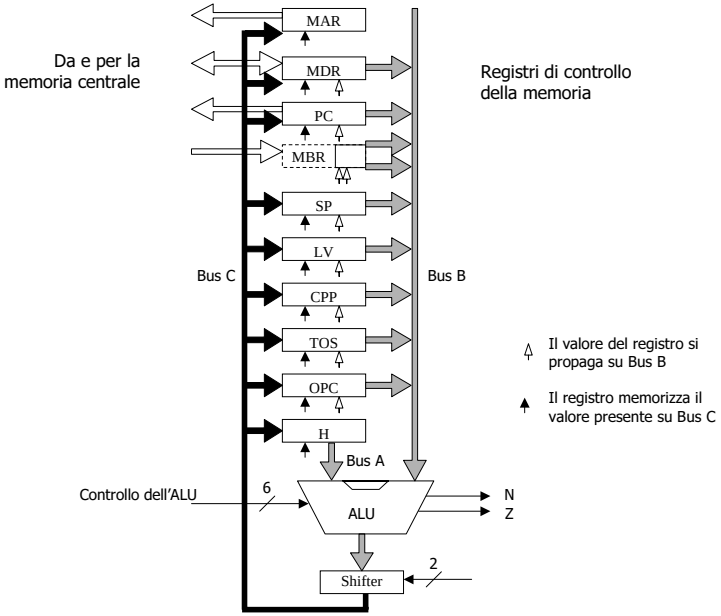
- Ogni istruzione prevede:
  - un codice operativo, che specifica se si tratta di un'operazione di ADD, SUB, BRANCH ...;
  - uno o più (eventuali) operandi;
- Ogni istruzione richiede:
  - uno (o più) cicli di clock;
  - accesso (eventuale) ad uno o più registri ed (eventuale) modifica del valore memorizzato.

Accesso alla memoria
(brevi richiami)

- La memoria può essere vista come una grande tabella dove:
  - le righe orizzontali sono dette parole;
  - tutte le parole hanno la medesima lunghezza (lunghezza di parola);
  - ogni parola è individuata da un indirizzo (o offset o scostamento);
  - la prima parola incontrata ha indirizzo 0;
  - con n bit si indirizzano 2^n parole (0 … 2^n-1)

Il data path

- E' quella parte di CPU che contiene l'ALU, con i suoi ingressi e le sue uscite.
- Un esempio di data path, della tipologia usata in gran parte dei calcolatori, è il seguente:



Osservazioni sul data path

- I registri sono tutti di 32 bit.
- L'ALU necessita degli ingressi BusA e BusB.
- Le 6 linee di controllo dell'ALU (F_0, F_1, ENA, ENB, INVA, INC) definiscono 64 possibili operazioni. Non sono pero' usate tutte:
  - ENA, ENB: abilitano l'entrata nell'ALU dei valori su BusA e BusB;
  - INVA: inverte l'ingresso su BusA;
  - INC: forza a 1 il carry-in.

Linee di controllo dell'ALU

F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\overline{A}$
1	0	1	1	0	0	$\overline{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B – A
1	1	0	1	1	1	B – 1
1	1	1	0	1	1	–A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
0	1	0	0	0	1	1
0	1	0	0	1	0	–1

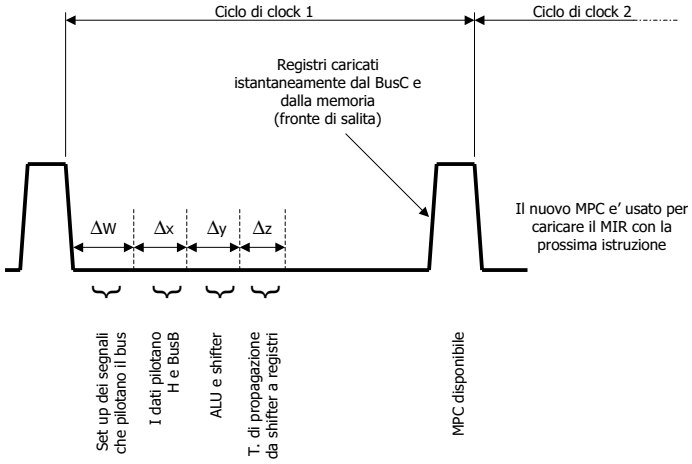
Lo shifter

- Le 2 linee di controllo dello shifter:
  - SLL8: Shift Left Logical di 8 bit. Sposta a sinistra di 8 bit, forzando a 0 gli 8 bit meno significativi;
  - SRA1: Shift Right Arithmetic. Sposta di un bit a destra senza modificare il bit più significativo.

Lettura e scrittura (stesso registro)

- Le operazioni di lettura e scrittura sullo stesso registro possono essere effettuate nello stesso ciclo di clock.
  - Es: SP = SP + 1
- Ciò richiede opportuna sincronizzazione fra gli elementi del data path.

Sincronizzazione nel data path



Sincronizzazione nel data path

- Le caratteristiche principali della sincronizzazione nel data path sono:
  - tutti i cicli di clock hanno la stessa durata;
  - i segnali di controllo si stabilizzano durante Δw;
  - i registri propagano i propri valori su BusB durante Δx;
  - ALU e shifter operano durante Δy;
  - i risultati si propagano su BusC durante Δz;
  - i registri memorizzano i valori osservati su BusC durante il fronte di salita.

Sincronizzazione nel data path

- **Osservazione**
  - I valori (e quindi gli operandi) sono letti sul fronte di discesa.
  - I valori (e quindi i risultati) sono scritti sul fronte di salita.

Sincronizzazione nel data path

- I progettisti del processore devono:
  - scegliere la frequenza di clock più alta possibile, per avere più cicli nella stessa unità di tempo e quindi disporre di un processore più veloce;
  - assicurarsi che l'intervallo $\Delta w+\Delta x+\Delta y+\Delta z$ accada con sufficiente anticipo rispetto al fronte di salita del ciclo di clock successivo: ciò limita il valore massimo della frequenza di clock.

Accesso alla memoria

- Vi sono due modalità di scambio dati tra memoria e processore:
  - word-addressable: il registro a 32 bit MAR (Memory Address Register) indirizza una parola di memoria e vi accede tramite il registro a 32 bit MDR (Memory Data Register);
  - byte-addressable: il registro a 32 bit PC (Program Counter) indirizza un byte di memoria e vi accede tramite il registro a 8 bit MBR (Memory Byte Register).

Accesso alla memoria

- Se si mette il valore 2 nel MAR e si effettua l'operazione di lettura, si troverà in MDR la parola 2 della memoria centrale, cioè i byte da 8 a 11 della stessa memoria centrale.
- Se si mette il valore 2 nel PC e si effettua l'operazione di lettura, si troverà negli 8 bit di MBR il contenuto del byte 2 della memoria centrale.

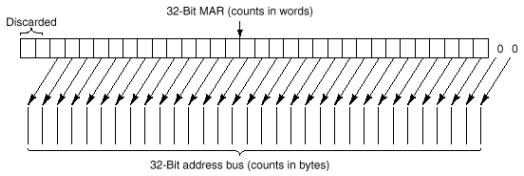
Accesso alla memoria

- MAR/MDR sono utilizzati per l'accesso ai dati del livello ISA.
- PC/MBR sono utilizzati per leggere il programma eseguibile del livello ISA.
- Tutti i rimanenti registri sono di 32 bit, e quindi word-addressable.

Accesso alla memoria

- Per passare da un conteggio in parole ad un conteggio in byte, bisogna moltiplicare per 4 (32 : 8 = 4).
- La moltiplicazione di un numero binario per 4 si ha aggiungendo semplicemente due zeri a destra dell'ultimo bit (analogamente, per moltiplicare per 10 un numero in base 10 si aggiunge uno 0 a destra dell'ultima cifra…)

MAR e bus degli indirizzi



Accesso alla memoria

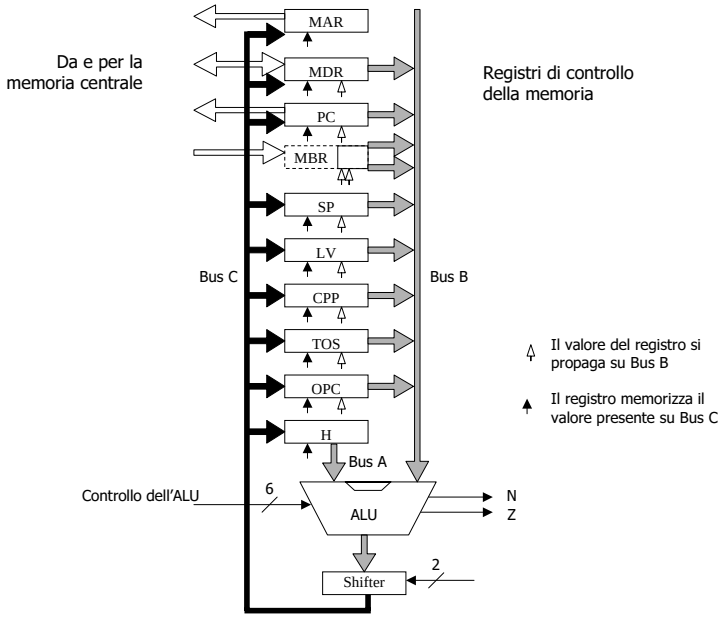
- La scrittura di MBR su BusB può essere effettuata considerando MBR come numero:
  - **unsigned** (0...255): su BusB, i 24 bit superiori (o piu' significativi) sono forzati a 0, gli 8 bit inferiori (o meno significativi) contengono il valore di MBR;
  - **signed** (-128…+127): su BusB, i 24 bit più significativi duplicano il segno di MBR, gli 8 bit meno significativi contengono il valore di MBR.
- I due segnali di controllo di scrittura di MBR specificano la scrittura unsigned o signed.

Il microprogramma

- E' composto da microistruzioni che controllano il data path.
- Le microistruzioni realizzano:
  - il fetch delle istruzioni assembler (cioè delle istruzioni fornite al livello ISA);
  - il decode delle istruzioni del livello ISA;
  - l'execute delle istruzioni del livello ISA.

Microistruzioni

- Una microistruzione è definita da (e specifica) l'insieme dei segnali di controllo nel data path per un certo ciclo di clock.
- Una microistruzione viene eseguita in un ciclo di clock.
- I possibili segnali di controllo nel data path sono 29 e riguardano:



Segnali di controllo del data path

- 9 per controllare la memorizzazione dei valori di BusC nei registri;
- 9 per controllare la propagazione su BusB dei valori dei registri;
- 8 per controllare le funzioni dell'ALU e dello shifter;
- 2 per indicare read/write attraverso MAR/MDR;
- 1 per indicare il fetch attraverso PC/MBR.

Durante ogni ciclo

- Sono letti i 29 segnali di controllo.
- Se è coinvolta un'operazione di lettura dalla memoria:
  i) nel ciclo corrente è definito il registro MAR;
  ii) nel ciclo successivo la memoria assegna i valori validi a MDR;
  iii) nel ciclo ancora successivo il valore letto dalla memoria in MDR viene utilizzato per l'operazione.
  (durante il secondo passo, il processore può effettuare altre operazioni purché non acceda alla memoria)

Accesso alla memoria

- Non sempre la memoria riesce a rispondere, ad una richiesta di accesso, in un solo ciclo di clock (avendo così un hit-rate del 100%).
- Per disporre di hit-rate elevati si usano memorie *cache* (vedi più avanti).

Microistruzioni

- Un **solo** registro *per volta* può propagare su BusB il proprio valore memorizzato.
- Più registri **contemporaneamente** possono memorizzare al proprio interno il valore osservato su BusC.
- Per codificare (e per decodificare) il controllo di BusB da parte di 9 registri sono necessari 4 bit.

Codifica delle microistruzioni

- Poiché un solo registro può scrivere su BusB, i segnali di controllo del data path sono:
  - 9 lettura da BusC +
  - 4 codifica assegnamento a BusB +
  - 8 controllo ALU e shifter +
  - 2 read/write +
  - 1 fetch =

  24 bit di controllo

Codifica delle microistruzioni

- Oltre ai 24 bit dei segnali di controllo, ogni microistruzione deve indicare

l'operazione da compiere nel ciclo successivo

- Tale informazione viene specificata attraverso due campi aggiuntivi.

Specifica della microistruzione successiva

- Avviene indicando nella microistruzione:
  - Addr: indirizzo della prossima microistruzione (a meno di salti);
  - JAM: JAMN, JAMZ e JMPC indicano i criteri per selezionare la prossima microistruzione (ad es. in base al risultato dell'ultima operazione effettuata dall'ALU che ha definito i valori delle uscite N e Z, memorizzate nei bistabili omonimi).

Specifica della microistruzione successiva

- Se i bit di JAM sono tutti 0:
  - Addr indica la prossima microistruzione. MPC copia il valore di Addr.
- Se JAMN o JAMZ sono diversi da 0:
  - la microistruzione successiva dipende dai valori memorizzati nei bistabili N e Z:
    - N = 1 indica che l'ultima operazione ha restituito un numero negativo;
    - Z = 1 indica che l'ultima operazione ha restituito un numero nullo (0).

Specifica della microistruzione successiva

- Se i bit di JAMN o JAMZ **NON** sono tutti 0, la microistruzione successiva (che come già detto dipende da N e Z) può essere esclusivamente:
  - il valore indicato da Addr;
  - il valore indicato da Addr + 256 (in realtà messo in OR con il numero esadecimale 0x100).

Specifica della microistruzione successiva

- Se JAMZ = 1:
  - se Z vale 0, la microistruzione successiva è Addr;
    - Es.: stiamo eseguendo la microistruzione 0x75, Z=0.

Indirizzo	Addr	JAM	Bit di controllo del data path
0x75	0x92	001	.....

la microistruzione successiva è Addr, cioè 0x92.

Specifica della microistruzione successiva

- Se JAMZ = 1:
  - se Z vale 1, la microistruzione successiva è Addr + 0x100;
    - Es.: stiamo eseguendo la microistruzione 0x75, Z=1.

Indirizzo	Addr	JAM	Bit di controllo del data path
0x75	0x92	001	.....

la microistruzione successiva è Addr+ 0x100, cioè 0x192.

Specifica della microistruzione successiva

- Se JAMN = 1:
  - se N vale 0, la microistruzione successiva è Addr;
    - Es.: stiamo eseguendo la microistruzione 0x75, N=0.

Indirizzo	Addr	JAM	Bit di controllo del data path
0x75	0x92	010	.....

la microistruzione successiva e' Addr, cioè 0x92.

Specifica della microistruzione successiva

- Se JAMN = 1:
  - se N vale 1, la microistruzione successiva è Addr + 0x100;
    - Es.: stiamo eseguendo la microistruzione 0x75, N=1.

Indirizzo	Addr	JAM	Bit di controllo del data path
0x75	0x92	010	.....

la microistruzione successiva è Addr+ 0x100, cioè 0x192.

Specifica della microistruzione successiva

- Se JMPC = 1:
  - si esegue l'OR bit a bit tra gli 8 bit meno significativi di Addr e MBR.
- Tale tecnica è molto utile per effettuare un salto calcolato (o jump a più vie), del tipo del costrutto C *switch*.
- Posso saltare alla microistruzione relativa all'opcode appena letto in MBR.

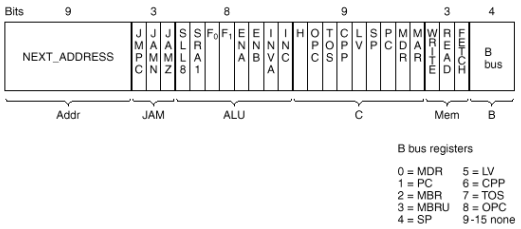
Esempio di uso di JMPC

- Istruzione al livello ISA: `istruz1`
  - Dopo il fetch, MBR = 0000 0001.
  - Se ADDR = 0 1010 1100, eseguendo l'OR tra gli 8 bit meno significativi di ADDR e MBR ottengo ADDR = 0 1010 1101
- Istruzione al livello ISA: `istruz2`
  - Dopo il fetch, MBR = 0000 0010.
  - Se ADDR = 0 1010 1100, eseguendo l'OR tra gli 8 bit meno significativi di ADDR e MBR ottengo ADDR = 0 1010 1110

Esempio di uso di JMPC

- Ottengo così una iniziale decodifica dell'istruzione, eseguendo:
  - la microistruzione di indirizzo 0 1010 1101 per la istruzione `istruz1` a livello ISA;
  - la microistruzione di indirizzo 0 1010 1110 per la istruzione `istruz2` a livello ISA.

Formato delle microistruzioni



I 36 bit sono spesso raggruppati in Addr (9), JAM (3), ALU (8), C (9), Mem (3), B (4).

Controllo delle microistruzioni

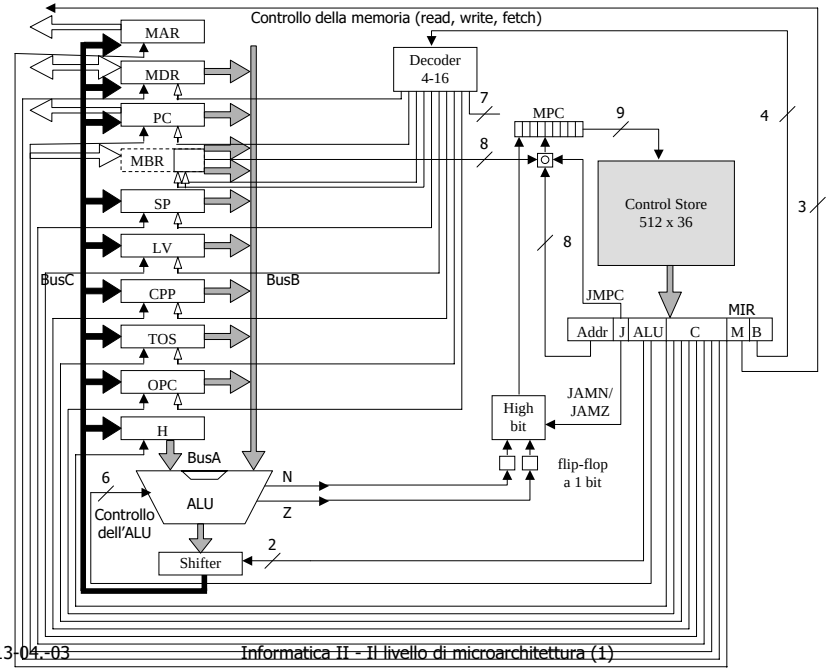
- L'emissione dei segnali di controllo durante ogni ciclo è effettuata da un sequenziatore, che in ogni ciclo basandosi sul contenuto di una memoria (Control Store, CS) provvede a definire:
  - lo stato di ogni segnale di controllo del sistema;
  - l'indirizzo della microistruzione seguente.

Il Control Store

- E' una memoria che include le microistruzioni (cioè i 36 bit del formato della microistruzione) di ogni passo della sequenza di controllo corrispondenti all'esecuzione di ogni istruzione dell'ISA.
- Ogni microistruzione specifica la microistruzione successiva.
- Il CS è dotato di 512 parole.

Il Control Store

- E' una memoria a sola lettura.
- Come tutte le memorie ha un:
  - registro di indirizzo: MPC (Micro Program Counter). Per indirizzare le 512 parole, MPC ha dimensione 9 bit;
  - registro dati: MIR (Micro Instruction Register). Per contenere le microistruzioni codificate, MIR ha dimensione 36 bit.



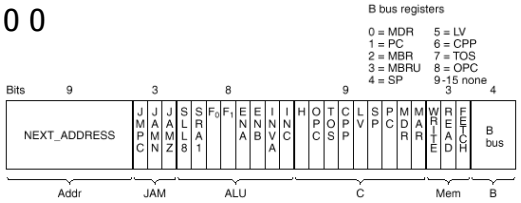
MPC

- MPC è un registro virtuale:
  - è possibile farne a meno;
  - gli ingressi a MPC possono essere collegati direttamente al Control Store;
  - la sincronizzazione risulta più semplice.

Microprogramma per Main1

- Main1: PC = PC + 1; fetch; goto (MBR)

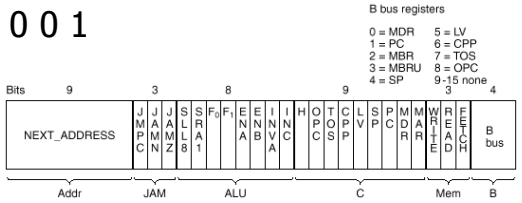
  legge in MBR la cella puntata da PC, incrementa PC; goto calcolato alla microistruzione indicata
  - NEXT_ADDRESS: goto calcolato (maschera)
  - JAM: 1 0 0
  - ALU: 0 0 1 1 0 1 0 1
  - C: 0 0 0 0 0 0 1 0 0
  - Mem: 0 0 1
  - B: 0 0 0 1



Microprogramma per IADD (i)

- IADD1: MAR = SP = SP - 1; rd

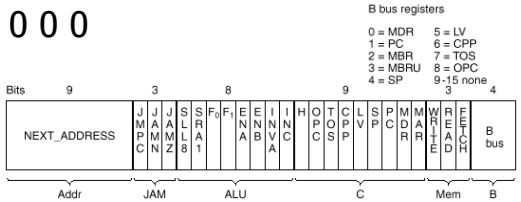
  legge la parola vicina alla cima dello stack
  - NEXT_ADDRESS: next
  - JAM: 0 0 0
  - ALU: 0 0 1 1 0 1 1 1
  - C: 0 0 0 0 0 1 0 0 1
  - Mem: 0 1 0
  - B: 0 1 0 0



Microprogramma per IADD (ii)

- IADD2: H = TOS;

  Assegna a H la cima dello stack
  - NEXT_ADDRESS: next
  - JAM: 0 0 0
  - ALU: 0 0 0 1 0 1 0 0
  - C: 1 0 0 0 0 0 0 0 0
  - Mem: 0 0 0
  - B: 0 1 1 1



Microprogramma per IADD (iii)

- IADD3: MDR = TOS = MDR + H; wr; goto Main1

  somma la cima dello stack con la precedente cima dello stack; scrive la cima dello stack; fetch
  - NEXT_ADDRESS: Main1
  - JAM: 0 0 0
  - ALU: 0 0 1 1 1 1 0 0
  - C: 0 0 1 0 0 0 0 1 0
  - Mem: 1 0 0
  - B: 0 0 0 0