SINTESI SEQUENZIALE SINCRONA

- IL DISPOSITIVO HA MEMORIA DEGLI EVENTI PASSATI (IN UN ARCO TEMPORALE FINITO) E SI BASA ANCHE SU DI QUESTI PER DETERMINARE GLI STATI SUCCESSIVI.

SI PROGETTANO COSÍ MACCHINE A STATI FINITI DETERMINISTICHE. LE F.S.M. SONO CARATTERIZZATE DA:

- FISICA REALIZZABILITÀ (STATI FINITI E NON DIPENDONO DA EVENTI FUTURI)
- DATO UNO STATO E UNA CONFIG, IL NUOVO STATO È IDENTIFICATO UNIVOCAMENTE.

UNA FSM È DEFINITA DA:

- I : ALFABETO IN INGRESSO
- U : ALFABETO D'USCITA
- S : INSIEME DEGLI STATI
- δ : FUNZIONE STATO PROSSIMO
- λ : FUNZIONE D'USCITA.

MACCHINA DI MEALY:

- λ COSTITUISCE LA RISPOSTA DELLA MACCHINA QUANDO, TROVANDOSI IN UN CERTO STATO PRESENTE, RICEVE UN SIMBOLO IN INGRESSO
- L'USCITA SI PRESENTA QUANDO LA MACCHINA CAMBIA STATO

MACCHINA DI MOORE:

- λ COSTITUISCE LA RISPOSTA DELLA MACCHINA ~~QUANDO~~, ALLO STATO IN CUI SI TROVA
- L'USCITA VIENE DATA QUANDO LA MACCHINA È IN UN DETERMINATO STATO.

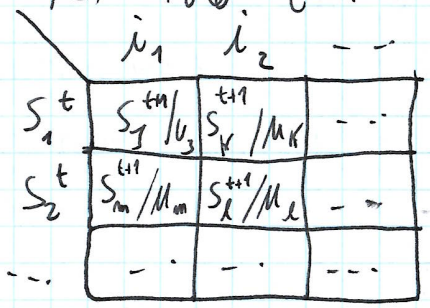
È POSSIBILE CONVERTIRE UNA MACCHINA DI MEALY IN UNA MACCHINA DI MOORE E VICE-VERSA.

SINTESI DI UNA FSM

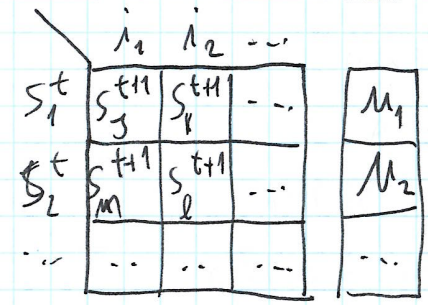
- IDENTIFICAZIONE DELLE FUNZIONI δ e λ
- SINTESI DELLE RETI COMBINATORIE CHE LE REALIZZANO.

LA SINTESI DELLA FUNZIONE δ DIPENDE DAI FLIPFLOP UTILIZZATI

LA FSM PUÒ ESSERE DESCRITTA TRAMITE UNA TABELLA:



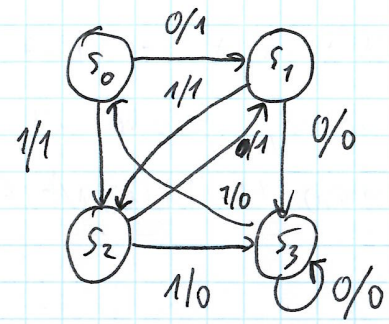
PER LA MACCHINA DI MEALY



PER LA MACCHINA DI MOORE

OLTRE ALLA TABELLA DEGLI STATI, È POSSIBILE REALIZZARE UN DIAGRAMMA DEGLI STATI, AD ESSA EQUIVALENTE.

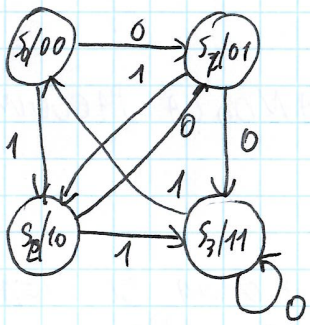
MEALY



=

	0	1
S_0	$S_1/1$	$S_2/1$
S_1	$S_3/0$	$S_2/1$
S_2	$S_1/1$	$S_3/0$
S_3	$S_3/0$	$S_0/0$

MOORE



=

	0	1	U
S_0	S_1	S_2	00
S_1	S_3	S_3	01
S_2	S_1	S_3	10
S_3	S_3	S_0	11

SINTESI DI UNA FSM: 2

DALLA TABELLA DEGLI STATI SI COSTRUISCE LA TABELLA DELLE TRANSIZIONI, ASSEGNANDO UNA CODIFICA AGLI STATI.

E.G. $S_0 = 00$, $S_1 = 01$, $S_2 = 10$, $S_3 = 11$ ⇒

	0	1	U
00	01	10	00
01	11	10	01
10	01	11	10
11	11	10	11

%/1

DALLA TABELLA DELLE TRANSIZIONI, SCELTI GLI ELEMENTI DI MEMORIA, SI PASSA ALLA TABELLA DELLE EQUAZIONI

RELÈ FLIP-FLOP SET-RESET

Q^t	Q^{t+1}	C	S	R
0	0	0	—	—
1	1	0	—	—
0	0	1	0	—
0	1	1	1	0
1	0	1	0	1
1	1	1	—	0

ATTUALE — DESTINAZIONE

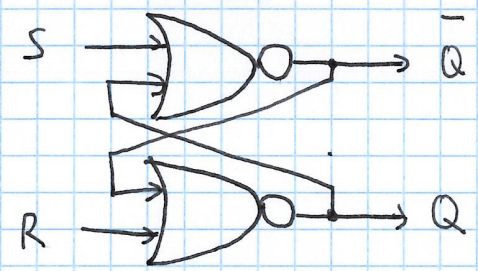
× 2 SR → SR, SR per ottenere Q^{t+1}

⇒ stati Q^t:

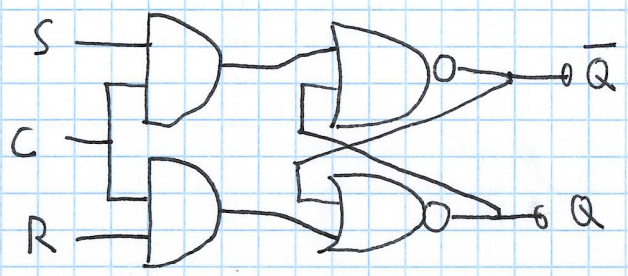
	0	1
00	0-, 10	10, 0-
01	10, 00	10, 01
10	01, 10	-0, 01
11	-0, -0	-0, 01

BISTABILI (ELEMENTI DI MEMORIA)

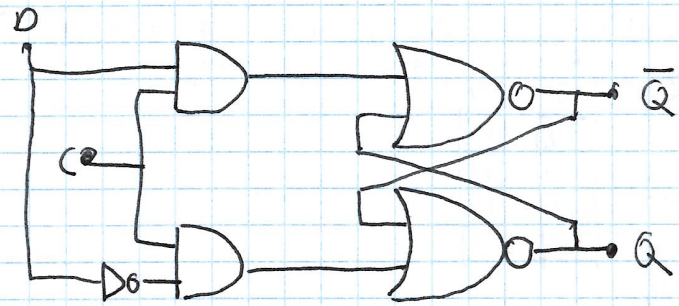
SET-RESET (BISTABILE ASINCRONO)



SET-RESET CON CLOCK (BISTABILE SINCRONO)



D-LATCH (SINCRONO)

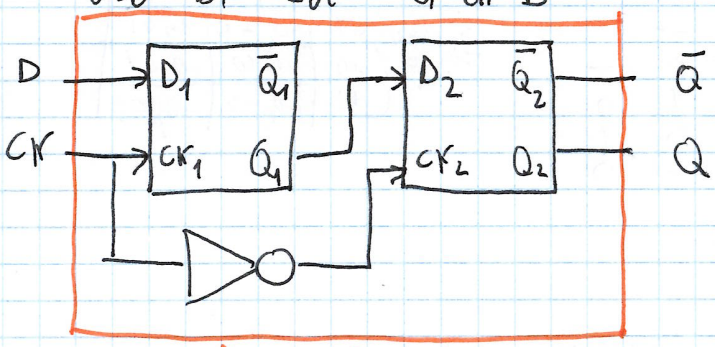


IL LATCH PRODUCE L'USCITA D QUANDO IL CLOCK È ALTO.

D-MASTER/SLAVE

S-R E D HANNO IL PROBLEMA DELLA TRASPARENZA DELLE USCITE QUANDO IL CLOCK È ALTO.

QUESTO PROBLEMA È RISOLTO DAL D-MASTER/SLAVE CHE FA USO DI DUE LATCH D



D-MASTER/SLAVE

SET DI OPERATORI FUNZIONALMENTE COMPLETI

- AND, OR, NOT
- AND, NOT
- OR, NOT
- NAND
- NOR

DE MORGAN

$$\overline{(\bar{A}+\bar{B})} = A \times B \qquad \overline{A+B} = \bar{A} \times \bar{B}$$

FORME CANONICHE

- 1ª FORMA CANONICA: SOMMA (OR) DI PRODOTTI (DEI TERMINI CHE RISULTANO 1) → SOMMA DI MINTERMINI

A	B	y
0	0	0
0	1	1
1	0	0
1	1	1

$\bar{A}B + AB$ } 1ª FORMA **SOP**

- 2ª FORMA CANONICA: PRODOTTO DI SOMME (DEI TERMINI CHE RISULTANO 0) → PRODOTTO DI MAXTERMINI

A	B	y
0	0	0
0	1	1
1	0	0
1	1	1

$(\bar{A}\bar{B}) \times (A\bar{B})$ } 2ª FORMA **POS**

KARNAUGH → OTTIMIZZA LA CARDINALITÀ DELLE PORTE A DUE LIVELLI (SOP)

- FORMULA $az + a'z = (a+a')z = z$
- METODO ESATTO
- APPLICARE DIRETTAMENTE LA FORMULA NON È SEMPRE IL MODO MIGLIORE

AD ESEMPIO

A	B	y
0	0	0
0	1	1
1	0	1
1	1	1

$y(A,B) = \bar{A}B + \bar{B}A + A\bar{B}$ → KARN. $= B(\bar{A}+A) + A\bar{B}$

$= B + A\bar{B}$

MA NON È LA FORMA MINIMA!

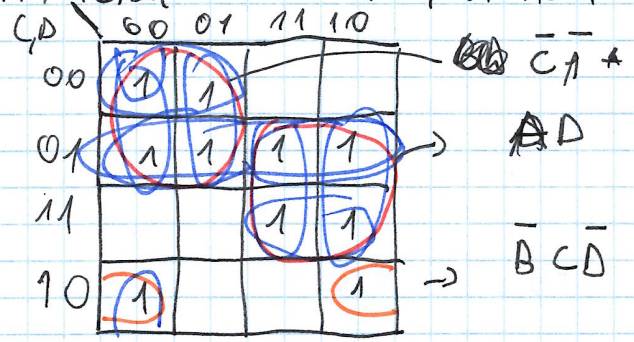
REPLICA BONE ←

$\bar{A}B + \bar{B}A + AB + AB \rightarrow A(B+\bar{B}) + B(A+\bar{A}) = A+B$

$X = X+X \Rightarrow$ ~~$X \neq X+X$~~

MAPPE DI KARNAUGH!

- CONSENTONO DI IDENTIFICARE CON PIÙ INTUIZIONE SIA I TERMINI DA RIDURRE CHE I TERMINI DA REPLICARE.
- APPLICABILE FINO A 4 VARIABILI



METODO DI QUINE-MCCLUSKEY: PRIMA FASE

0001	1		-000	1,9 ESSENZIALE
1001	9		10-1	9,11
1100	12		1-01	9,13
			110-	12,13
1011	11		11-0	12,14
1101	13			
1110	14		1-11	11,15
			11-1	13,15
1111	15		111-	14,15

1--1 9,11,13,15
11-- 12,13,14,15

SI CONFRONTANO I MINTERMINI CHE DIFFERISCONO DI UN UNO.

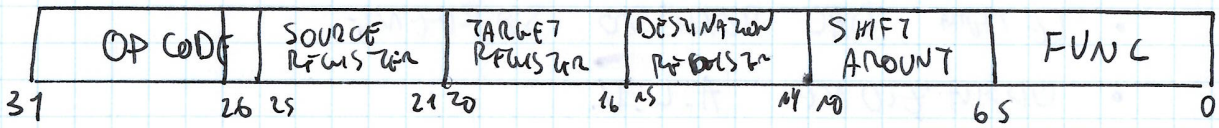
SI CONFRONTANO I RISULTATI CON DONCARE NELLA STESSA POS.

METODO DI QUINE-MCCLUSKEY: SECONDA FASE

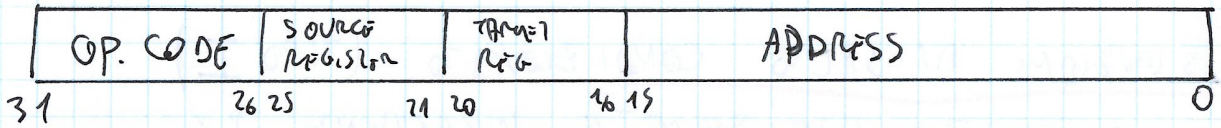
SI UTILIZZA LA TABELLA DI COPERTURA.

- SI RIDUCE LA TABELLA TRAMITE CRITERI DI ESSENZIALITÀ E DOMINANZA.
- A TABELLA IRRIDUCIBILE SI USANO CRITERI DI BRANCH AND BOUND.

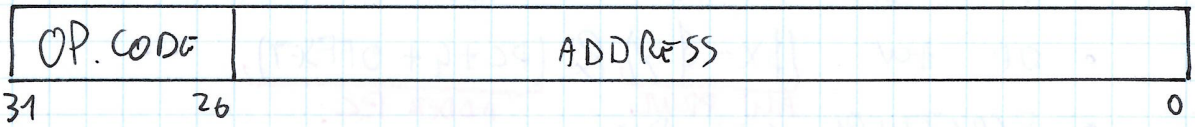
ISTRUZIONI DEL MIPS



R ?



I IMMEDIATE



JUMP

ISTRUZIONI DI TIPO R

LE ISTRUZIONI DI TIPO R, ARITMETICO - LOGICHE, VENGONO ESEGUITE IN 4 PASSI:

- PRELIEVO ISTRUZIONE E INCREMENTO DEL P.C.
- CARICAMENTO DATI DAI REGISTRI SORGENTE
- ALU
- SCRITTURA NEI REGISTRI DEL RISULTATO

ISTRUZIONI DI TIPO I (LOAD/STORE)

LE ISTRUZIONI DI LOAD/STORE, DI TIPO I, VENGONO ESEGUITE IN 5 PASSI:

- PRELIEVO ISTRUZIONE E INCREMENTO DEL P.C.
- LETTURA REGISTRO BASE
- OP ALU (BASE + OFFSET)
- PRELIEVO DATO DALLA MEMORIA
- SCRITTURA DATO IN UN REGISTRO DESTINAZIONE.

OPERAZIONI ARITMETICHE IMMEDIATE (DI TIPO I)

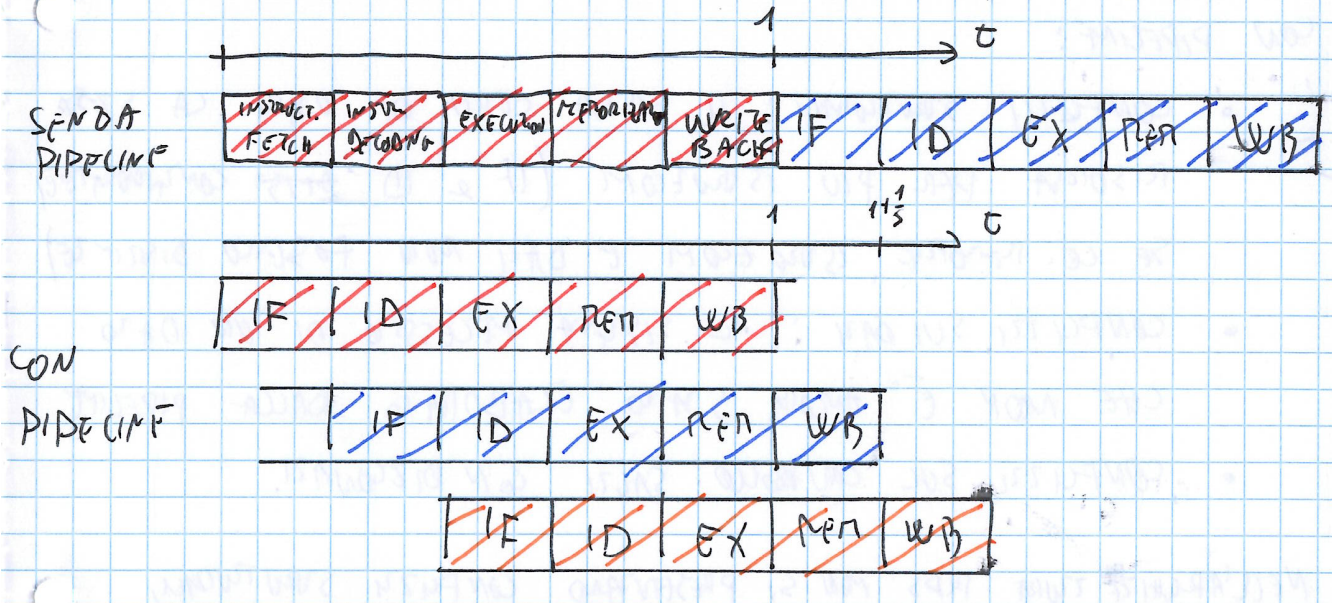
- PRELIEVO ISTRUZIONE E INCREMENTO P.C.
- LETTURA DAL REGISTRO SORGENTE
- OPERAZIONE A.L.V.
- SCRITTURA NEL REGISTRO DESTINAZIONE

ISTRUZIONI DI SALTO CONDIZIONATO (TIPO I)

- PRELIEVO ISTRUZIONE E INCREMENTO P.C.
- LETTURA DEI DUE REGISTRI SORGENTI
- OP ALU ($X - $Y) & (PC + 4 + OFFSET)
  ALU PRINC. ADDER P.C.
- SCRITTURA NEL P.C.

PIPELINING

NON RIDUCE IL TEMPO PER ISTRUZIONE, MA AUMENTA IL THROUGHPUT.



NON TUTTE LE ISTRUZIONI RISC NECESSITANO DI 5 PASSI PER ESSERE ESEGUITE. ANZI, ESCLUSIVAMENTE LA LOAD NECESSITA DI 5 PASSI. QUESTO PORTA AD UN INEVITABILE PERDITA DI PRESTAZIONI NEL TEMPO DI ESECUZIONE DI UNA SINGOLA ISTRUZIONE. DI CONTRO, IL THROUGHPUT SI QUINTUPLICA PER UNA PIPELINE A 5 STADI.

SI INTRODUCONO I REGISTRI DI PIPELINE CHE CONSENTONO LO SCAMBIO DI DATI TRA LE COMPONENTI HARDWARE COINVOLTE NEI DIVERSI STADI. CASO SPECIALE È QUELLO DEI SEGNALI DI CONTROLLO. ANCHESSI VENGONO TRASPORTATI TRA I VARI STADI HARDWARE.

PIPELINE HAZARDS

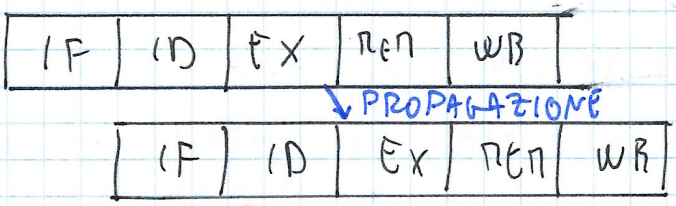
SI POSSONO SVILUPPARE 3 TIPI DI CONFLITTI IN UN SISTEMA CON PIPELINE:

- CONFLITTI STRUTTURALI: IL SISTEMA CERCA DI USARE LA STESSA RISORSA PER PIÙ ISTRUZIONI (IF e ID ~~DATI~~ CONTEMPORANEE, SE CE MEMORIE ISTRUZIONI E DATI NON FOSSERO DISTINTE)
- CONFLITTI SUI DATI: IL SISTEMA NECESSITA DI UN DATO CHE NON È ANCORA STATO ELABORATO DALLA PIPELINE
- CONFLITTI SUL CONTROLLO: SALTI CONDIZIONATI.

NELL'ARMITE TUTTI MIPS NON SI PRESENTANO CONFLITTI STRUTTURALI, DATA LA SEPARAZIONE DELLA MEMORIA ISTRUZIONI DALLA MEMORIA DATI E L'ACCESSO COERENTE AL BANCO DEI REGISTRI.

CONFLITTI DI DATI

```
add $s0, $t0, $t1
sub $t2, $s0, $t3
```



SI USA LA PROPAGAZIONE TRA EX/MEM DELLA ADD E ID/EX DELLA SUB.

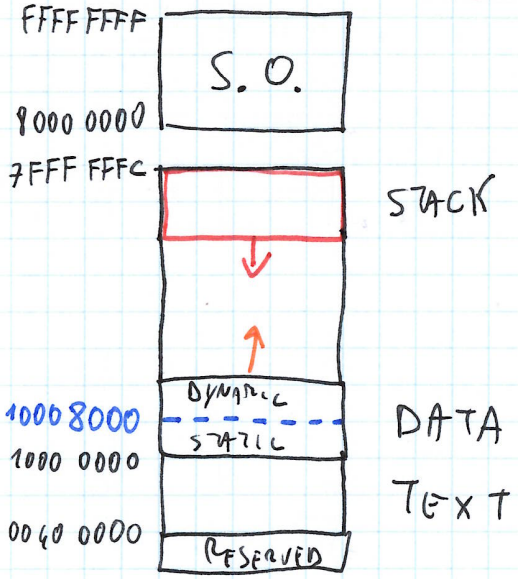
NEL CASO DI UNA LOAD + OPERAZIONE NON È PERÒ POSSIBILE PROPAGARE IL DATO, PERCHÉ QUESTO DIVENTA DISPONIBILE NELLO STADIO MEM/WB. IN QUESTO CASO SI INSERISCE UNA BOLLA, O NOP, PER RITARDARE UN'ISTRUZIONE RISPETTO ALLA SUCCESSIVA, E RENDERE POSSIBILE LA PROPAGAZIONE.

CONFLITTI DEI CONTROLLI

beq OPERANDI CONFRONTO → BRANCH PREDICTION: SE LA PREDIZIONE È SBAGLIATA, FLUSH DELLA PIPELINE

→ PROPAGAZIONE DEGLI OPERANDI A UN COMPARATORE SEPARATO CHE NELLO STADIO ID

PARTIZIONE DELLA MEMORIA



DICHIARAZIONE DI VAR GLOBALI

```
.data <addr>     // LA MEMORIA PARTIRÀ DA ADDR
.asciiz "str"    // STRINGA CON \0
.ascii  "str"    // STRINGA SENZA \0
.byte b1,b2...bn // ARRAY DI BYTE INIZIAL. A b1,b2,bn
.WORD w1, .. wn  // ARRAY DI WORD
.space n         // ALLOCA n BYTE
.text <addr>     // MEMORIA ISTRUZIONI
.globl sym       // ETICHETTA sym
.eqv             // come #define in C
```

SYSTEM CALL

È POSSIBILE EFFETTUARE CHIAMATE DI SISTEMA, PASSANDO IL CODICE DELLA CHIAMATA NEL REGISTRO $V0 E CHIAMANDO IL COMANDO 'syscall'.

CODICI DELLE SYSCALL

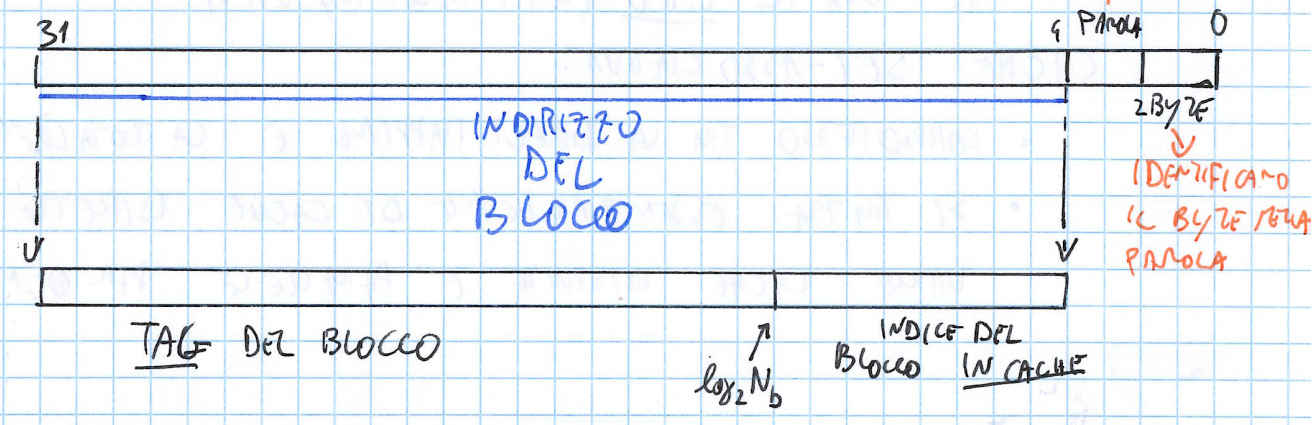
NOME	COD	ARG	RES
print_int	1	$a0	/
print_float	2	$F12	/
print_double	3	$F12	/
print_string	4	$a0	/
read_int	5	/	$v0
read_float	6	/	$F0
read_double	7	/	$F0
read_string	8	$a0,$a1	/
sbrk	9	$a0	$v0
	10	/	/

→ $a0 È UN BUFFER DI CARATTERI SPECIFICATA DA $a1

MEMORIE CACHE

CACHE DIRECT - MAPPED

SE LA CACHE PUÒ CONTENERE N_b BLOCCHI, $\log_2 N_b$ È IL NUMERO DI BIT RICHIESTI PER INDIRIZZARLI.



OGNI BLOCCO DI CACHE AVRÀ QUINDI:

- 1 BIT DI VALIDITÀ
- UN CAMPO ETICHETTA
- UN CAMPO CONTENUTO (DATI)

IL BLOCCO DI CACHE CONTIENE TIPICAMENTE 8-16 PAROLE.

SCRITTURA

SI POSSONO ADOTTARE DIVERSE SOLUZIONI PER MANTENERE LA SINCRONIA TRA CACHE E RAM:

- WRITE-THROUGH: TUTTE LE SCRITTURE VENGONO EFFETTUATE SIMULTANEAMENTE SIA IN CACHE CHE IN RAM. QUESTO APPROCCIO È LENTO.
- WRITE BUFFER: CACHE E WRITE BUFFER RICEVONO SIMULTANEAMENTE LA SCRITTURA, IL CONTROLLORE DEL WRITE BUFFER PROPAGA POI LA MODIFICA IN RAM, INDIPENDENTEMENTE DALLA CPU. 4-8 POSIZIONI
- WRITE BACK: IL BLOCCO MODIFICATO VIENE SCRITTO IN RAM SOLO QUANDO DEVE ESSERE SOSTITUITO. SI SCRIVE SOLO UNA VOLTA IN RAM, MA LA MISS PENALTY RADDOPPIA
- WRITE BACK CON DIRTY BIT: SI TIENE UN BIT DI "DIRTY" CHE INDICA SE IL BLOCCO È STATO MODIFICATO. SE NON È STATO MODIFICATO, NON C'È BISOGNO DI SCRIVERLO IN RAM.

ALCUNE SOLUZIONI CACHE

CACHE TOTALMENTE ASSOCIATIVA:

- QUALSIASI BLOCCO PUÒ ANDARE OVUNQUE.
- IL TAG È L'INDIRIZZO DEL BLOCCO
- NECESSITA DI UNA POLITICA DI SOSTITUZIONE. TIPICAMENTE SI USA IL LRU (LEAST RECENTLY USED)

CACHE SET-ASSOCIATIVA:

- COMPROMESSO TRA LA DIRECT MAPPING E LA TOTALMENTE ASSOCIATIVA
- SI TRATTA ESSENZIALMENTE DI CACHE DIRETTE CON VARIE CACHE DISTINTE E PARALLELE PER OGNI TAG.

MEMORIA VIRTUALE

- PIÙ PROGRAMMI IN ESECUZIONE MEMORIA SIMULTANEAMENTE DEVONO AVERE SPAZI DI INDIRIZZAMENTO DISTINTI (PROTEZIONE)
- LA MEMORIA VIRTUALE FA CIÒ CHE C'È UNA CACHE E RAM TRA DISCHI MAGNETICI E RAM.
- UN SINGOLO PROGRAMMA PUÒ OCCUPARE PIÙ MEMORIA PRIMARIA DI QUELLA DISPONIBILE.

LA MEMORIA PRINCIPALE È CHIAMATA MEMORIA FISICA, E I SUOI INDIRIZZI SONO CHIAMATI INDIRIZZI FISICI

LA MEMORIA VIRTUALE HA INDIRIZZI RILOCABILI (PARTONO DA 0) E OGNI SPAZIO DI INDIRIZZAMENTO È DEFINITO DAL N° BIT DEGLI IND.

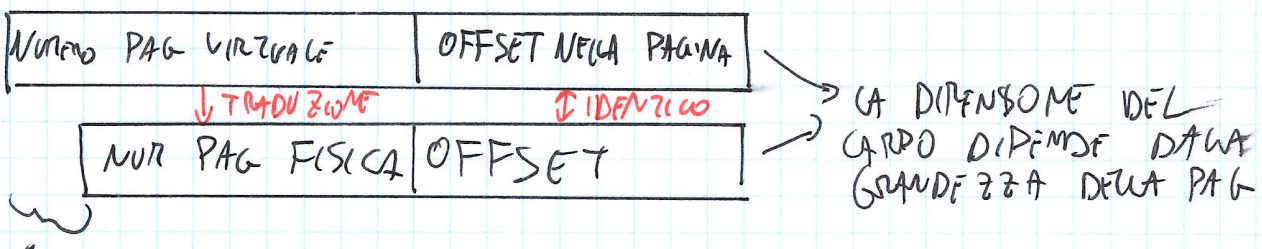
C'È UN MECCANISMO AUTOMATICO DI TRADUZIONE TRA MEM. VIRTUALE E INDIRIZZI FISICI. (RILOCAZIONE DINAMICA)

QUESTO SISTEMA È TRASPARENTE AL PROGRAMMATORE, AL COMPILATORE E AL LINKER.

LA MEMORIA VIRTUALE È ORGANIZZATA IN PAGINE.

SI POSSONO VERIFICARE PAGE FAULT

INDIRIZZO VIRTUALE:



Δ

TOTALMENTE ASSOCIATIVA + CAST (RECENTLY USED) PER LA SOSTITUZIONE MA SI POSSONO ANCHE USARE ALGORITMI COMPLESSI, A CARICO DEL S.O. AL POSTO DELLA CPU

PER TRADURRE DA N° PAG. VIRT A N° PAG. FIS. SI USA UNA TABELLA DELLE PAGINE

TABELLA DELLE PAGINE

SI USA UNA TABELLA DELLE PAGINE PER OGNI PROCESSO.
IL NPV PUÒ QUINDI ESSERE USATO COME INDICE AL FINE DI RECUPERARE IL NPF.

PANELLA REALTÀ È IMPOSSIBILE ESPLICITARE UNA TABELLA NELLA MEMORIA FISICA. ANCHE LE TABELLE SONO SOGGETTE A PAGINAZIONE.

MEMORY MANAGEMENT UNIT

DISPOSITIVO HARDWARE CHE HA IN CARICO LA GESTIONE DELLE PAGINE DI MEMORIA.
IN PARTICOLARE, CONTIENE UNA MEMORIA "TLB" IN CUI ASSOCIA PID + NPV ⇔ NPF.

SE LA PAGINA NON È TROVATA NEL TLB, LA RICERCA VIENE EFFETTUATA NELLA TABELLA DELLE PAGINE. SE NEANCHE LÌ È PRESENTE, SI GENERA UN PAGE FAULT