

lista a latentelor interzise \rightarrow tr. 1: t_0 ; tr. 2: t_1 ; tr. 3: t_2 ; tr. 4: t_3 ; tr. 5: t_4

\rightarrow vector de coliziune: 5 elem.:

$c_4 \ c_3 \ c_2 \ c_1 \ c_0$

$(1 \ 0 \ 0 \ 1 \ 1)$

$4 \ 3 \ 2 \ 1 \ 0 = \text{latentă}$

\rightarrow

\rightarrow conține latentele interzise.

\rightarrow se generează diagrama stărilor: pg. 123:

- se pleacă din starea dată de vect. de coliziune;
- urm. stare se obț. deplasând spre dre. cu **2** ~~vector~~ starea anterioară și făcând ~~sau~~ sau între rez. și vectorul de coliziune (cel inițial);

asta la pasul 2, pentru că
aveam latentă 2; la pas
3 shiftăm dre. cu 3 poz. (latentă = 3) \rightarrow pg. 123

\rightarrow latentă medie minimă (MAL) = \min (latentele medii)

\downarrow
ciclul cu MAL maximizează rata de transfer ppr. ppl.

\rightarrow latentă minimă \rightarrow ar fi necesar de un mec. care să numere (2 latentă, 3 latentă) și să introducă intri după acele latente ($i_1 \ u_1 \ i_2 \ u_2 \ i_3 \ u_3 \ i_4 \ u_4 \ i_5 \dots$) ast. MAL = (2,5); ~~se poate~~ e mai simplu să avem un mec. care introduce mereu intri cu aceeași latentă \rightarrow latentă minimă: $i_1 \ u_1 \ i_2 \ u_2 \ i_3 \ u_3 \ i_4 \dots$
 \rightarrow cu cadenta asta utza. scade de 3 ori, dar ppl. fct. simplu (pg. 129)

\rightarrow tot ce e valabil ppr. ppl. static e valabil și ppr. ppl. dinamic

\rightarrow pg. 30: 4 liste de latente și o intr. de coliziune \rightarrow

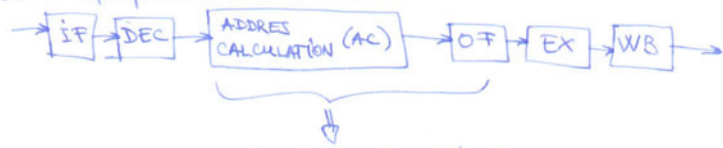
\downarrow
AA, BB, AB, BA

\downarrow
~~MA~~ $M_A \ \& \ M_B$

\rightarrow diagrama stărilor (pg. 125)

pg. 135 - diagrama stărilor \Rightarrow la fel ca la struct. statică, dar
ptr. intr. de coliziune... e posibil să 3 me. greșate m. vite...

pg. 129: Pip. ptr. instr. cu 6 trepte:



OF de la pip. cu 5 trepte
s-a decompus în 2 trepte

Obs. că \times posibilitate de coliziune (ptr. că 3 reacție)

IF, OF & WB sunt op. consumatoare de timp \Rightarrow ca să se poată
face o planificare corectă, aceste op. sunt asoci. cu întârzieri, aș.
să se obț. o cadență de intrare a instr. în pip. destul de
mare aș. să permită terminarea op. consumatoare de timp ptr.
instr. anterioară (să se scrie corect în memorie).

Între DEC & AC & între EX & WB nu mai apar întârzieri.

Dezavantaj \Rightarrow ~~instr.~~ arh. mai complicată (14 trepte)

\Rightarrow instr. necesită un mult timp ptr. umplerea struct.
pip. la pr. instr. (14 clocks) \Rightarrow ! fa instr. de salt!

Avantaj \Rightarrow scot instr. din pip. de 3 ori mai repede.

branch prediction
e fr. timp.

pg. 141: Procesor aritmetic \Rightarrow tipic pip.

pip. cu 5 trepte la adunare \Rightarrow frev. mică cu care instr.
parăsese pip. (op. consumă de
timp)

pip. cu 8 trepte la adunare \Rightarrow frev. mai mare

Manumite bit. se introduc reacții \rightarrow de ex. la ^{prima} deplasare, op.
durează un nr. variabil de perio. de ceas, m. faste. de rez. scan-
dare exponențială \rightarrow facem op. de depl. pe 1 bit cu reacție


Normalizare \Rightarrow detectează Δ și deplasează virgula cu 1 la
fie pe pr. poz.

Și după rotunjire e nevoie de normalizare.

La înmulțire $\begin{cases} 5 \text{ trepte} \\ 6 \text{ trepte} \end{cases}$

\rightarrow produse parțiale: reactive, ptr. că aceste
produse sunt în nr. variabil

pg. 160: CISC vs. RISC:

- \rightarrow programarea ^(p) la nivel mult a impus dezvolt. tehnologiei \Rightarrow op. sup. au
fost implementate direct în hws. \rightarrow arh. stufoase
- \rightarrow RISC \Rightarrow instr. complexe sunt înloc. cu succesiuni de instr. mai
simple \rightarrow hws. mai simple \rightarrow mai ieftin și mai repede disponibil
 \rightarrow viteză mai mare a accesului \rightarrow deș.
f. m. multe instr., ele se exec. m.
repede ptr. că ceasul e mai rapid ^{integrat}
- \rightarrow CISC \rightarrow CPU ocupă 40%-60% din chip, în urma tehnicilor de ~~integrare~~
- \rightarrow RISC \rightarrow  10% \rightarrow mai mult sp. ptr.
hws. pe chip \rightarrow m. multă memorie on-chip \rightarrow f. rapi-
dă (registre m. multe + cache mai mare)

avantaj: reg.
alcat. de la
Pentium

nu pot fi (+) de multe, ptr. că reg. tb.
să aibă nume \rightarrow de. de ex. un reg. are
nume de 6 bti, mai rămân ~~2~~ 2 bti
din octet ptr. codifi. op. \rightarrow tb. să fie
2 octeti ptr. instr. \rightarrow durată mai
multă să citească din memorie, și arunde

Cap. 3: Rețele de Interconectare:

Nod = procesor \pm memorie \pm I/O

Switch \Rightarrow abstr. leg. între o mulțime de in. și o mulțime de out., în mod
dinamic \Rightarrow simultaneitate a m. multor legături.

rearrangeable \Rightarrow In an unwrite circumstance pot să lăg i lăg'...

Topologie $\begin{cases} \text{statique} \rightarrow \text{legs fixe} \\ \text{dynamique} \end{cases}$

Retele statice:

- gradul unui nod: nr. de legi, pe care nodul le are cu lumea-externă;
- gradul unei rețele e maximul gradului nodurilor care compun rețeaua;
- rețele cu grad: 1: magistrală comună
2: linear array, reg.
3: binary tree, fat tree, shuffle-exchange
4: 2 dimensional mesh (ilicac, torus)
N: N-cube, N-dimensional mesh, k-ary N-cube
- diametrul rețelei: cea mai mare distanță minimă între (4) pereche de noduri

(distanța între 2 noduri = nr. de legi ce trebuie parcurse de la nodul sursă la nodul dest.)

- mag. comună: (pg. 6) Nu se poate lega la un nod decât numai la un alt nod N_j
- faptul că \exists o sg. mag. face imposibilă simultanizarea legi. $N_0-N_1 \text{ \& } N_2-N_3 \Rightarrow$ conflicte fiducie
 - (Hb. să \exists un arbitru al mag.)
 - diam. = 1

- linear array (pg. 7):
- gradul = 2 (†) nod are 2 legi, max puțin
nodurile de început și sfârșit

ca 12 scap 21
de degenantje

- ptr. N noduri, diam. e $N-1$
- apar intergrat în comunicație (mari)
- dezavantaj: doi se rupe o leg., avem 2 rețele autonome

- ring (p. 8): • toate nodurile au gradul 2

- avantaj: scalability (pot să adaugi uxor niveluri)
- pot să o implem. cu large scale integration → va fi f. ușor să adaugi niveluri
- aplicație tipică → procesoare pe 1 bit → fol. în pb. de inteligență artificială (decizie de genul: if true → ia calea x else → ia calea y | ⇒ generarea traiectoriilor ptr. roboți, etc.)
- pb.: fiabilitatea e dată de nr. de leg. → de se rupe o leg. se obț. 2 rețele independente nodurile sunt din ce în ce mai congestionate pe măsură ce avansăm spre rădăcină și la fel și leg. de comunic. → pb.: mai multe mesaje așteptate

→ fat tree (pg. 11): ramurile devin mai "groase" spre rădăcină ⇒ diminuează congestia (∃ m. multe posib. de comunicare între noduri de pe niș. dif.)

- se complică hwr. → hb. nec. de decupie (pe ce leg. mă duc de la N_1 la N_i ?)

→ shuffle-exchange (pg. 13): se utilizează 2 faze: shuffle & exchange

- shuffle: ~~shift~~ roție cu o poz. binară la stg., numele binar al unei combinații de $(S_{n-1}, S_{n-2}, \dots, S_0) \rightarrow (S_{n-2}, \dots, S_0, S_{n-1})$
- exchange: complementează LSB al șirului de biți ce îi este argument: $e(S_{n-1}, \dots, S_1, S_0) \rightarrow (S_{n-1}, \dots, S_1, \bar{S}_0)$
- utiliz. de alg. ll ptr. tr. Fourier rapidă (FFT), sortare, transpunere matrice, etc.