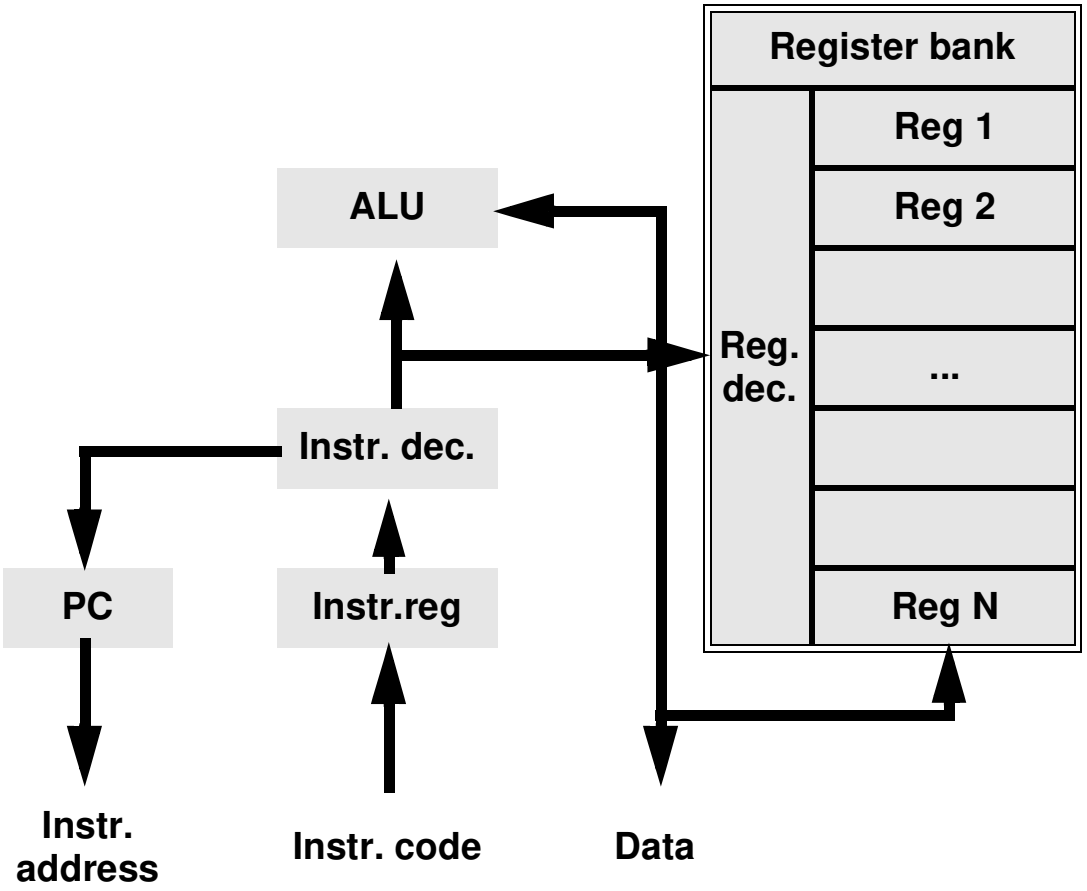
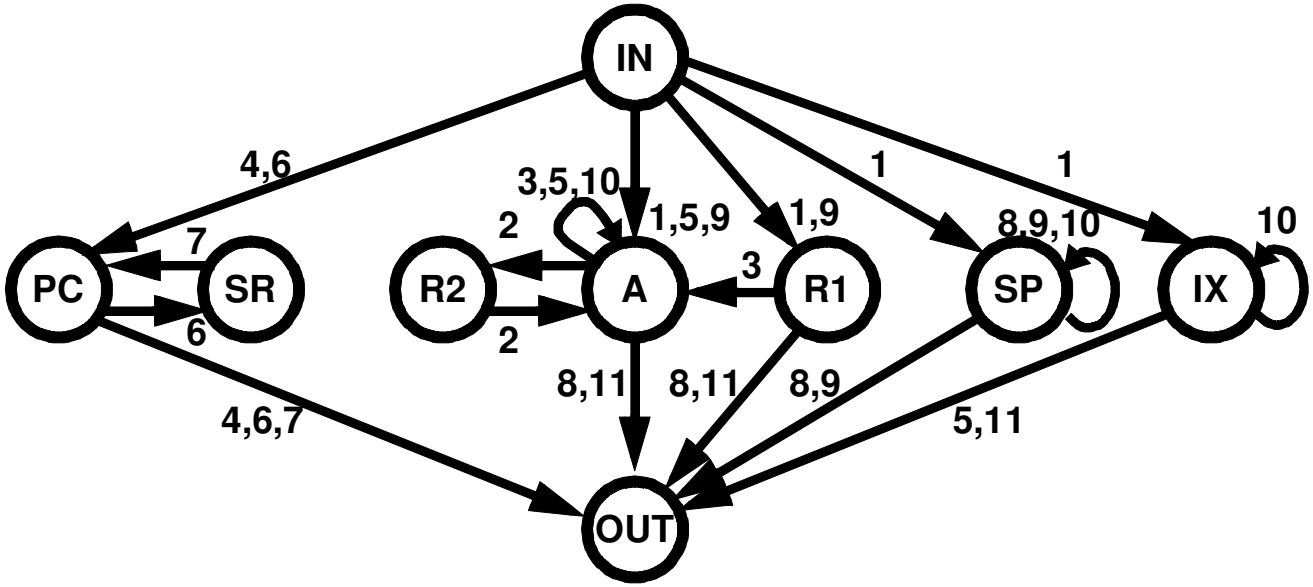
CPU fault models and test algorithms



Functional fault model

- **Register as memory, incl. addressing faults**
- **Instrucion decoder:**
  - **No instruction activated (NOP)**
  - **Another instrucion activated**
  - **Additional instrucion activated**
- **Refined model for microprogramed CPU**
  - **Instrucion: a sequence of microinstrucions**
  - **Microinstrucion: parallel executed microorders**
  - **Fault:**
    - **One or multiple microorders inactive**
    - **Faulty activation of microorders**
    - **Addititional microorders active**
    - **Another microorders active**
    - **Another microinstrucion activated (independently or linked)**
    - **Additional microinstrucion activated**
- **Data transfer: "logic transfer path"**
- **Data manipulation: from the structural model**

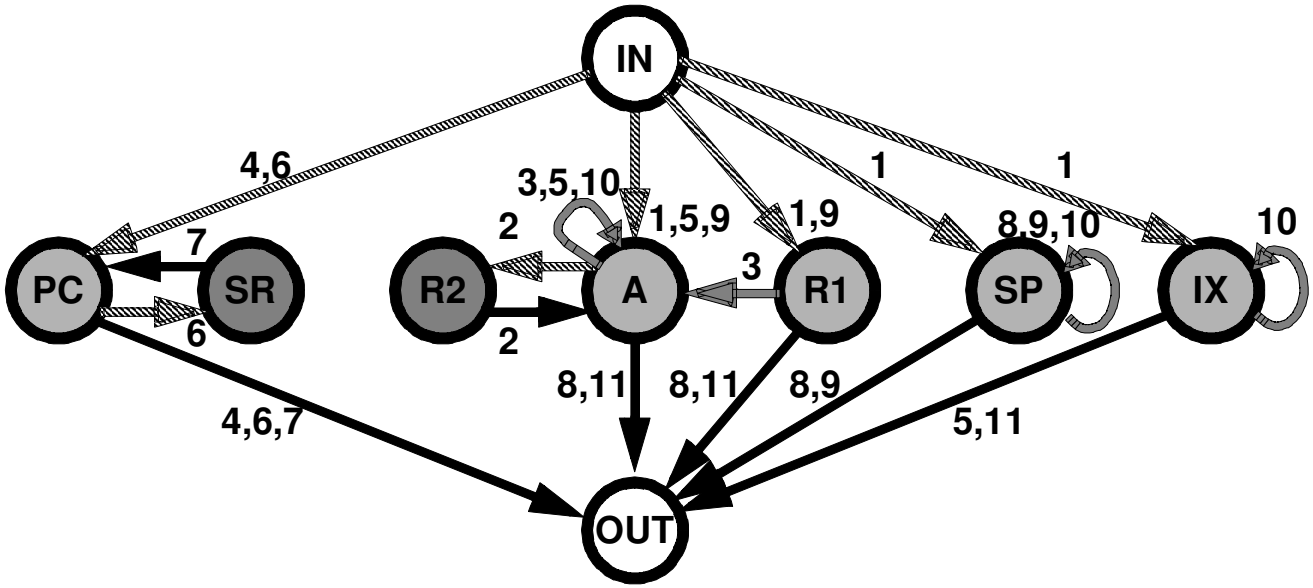
S-graph model



No	Instruction	Operation	Edge
1	MVI R,a mit $R \in \{A, R1, SP, IX\}$	$R \leftarrow a$	$IN \rightarrow R$
2	$MVI\ Ra, Rb$ mit $Ra, Rb \in \{A, R2\}$	$Ra \leftarrow Rb$	$Ra \rightarrow Rb$
3	$ADD\ A, R1$	$A \leftarrow A + R1$	$R1 \rightarrow A\ ,\ A \rightarrow A$
4	JMP a	$PC \leftarrow a$	$IN \rightarrow PC$ $PC \rightarrow OUT$
5	$ADD\ A, (IX)$	$A \leftarrow A + (IX)$	$IX \rightarrow OUT$ $IN \rightarrow A\ ,\ A \rightarrow A$

No	Instruction	Operation	Edge
6	$CALL$ a	$SR \leftarrow PC$ $PC \leftarrow a$	$PC \rightarrow SR$ $IN \rightarrow PC$ $PC \rightarrow OUT$
7	RET	$PC \leftarrow SR$	$SR \rightarrow PC$ $PC \rightarrow OUT$
8	PUSH R mit $R \in \{A, R1\}$	$\dot{SP} \leftarrow R$ $\dot{SP} \leftarrow \dot{SP} + 1$	$SP \rightarrow OUT$ $R \rightarrow OUT$ $SP \rightarrow SP$
9	POP R mit $R \in \{A, R1\}$	$R \leftarrow \dot{SP}$ $\dot{SP} \leftarrow \dot{SP} - 1$	$SP \rightarrow SP$ $SP \rightarrow OUT$ $IN \rightarrow R$
10	INC R mit $R \in \{A, SP, IX\}$	$R \leftarrow R + 1$	$R \rightarrow R$
11	$MOV\ (IX), R$ mit $R \in \{A, R1\}$	$(IX) \leftarrow R$	$IX \rightarrow OUT$ $R \rightarrow OUT$

Test of the register-decoders: R/W instruction sequences for each register



Register	$WRITE_R$	$READ_R$	Label(R)
OUT	-	-	0
PC	*JMP* a	*CALL* a	1
SR	$WRITE_{PC}$ + *CALL* a	RET + $READ_{PC}$	2
R2	$WRITE_A$ + *MVI A, R2*	*MVI R2, A* + $READ_A$	2
A	*MVI* A,a	*MOV (IX), A*	1
R1	*POP* R1	*MOV (IX),* R1	1

Register	$WRITE_R$	$READ_R$	Label(R)
SP	*MVI SP, a*	*POP* R1	1
IX	*MVI (IX), a*	*MOV (IX), R*	1

Test of the register decoding

- **To be proven:** $\forall i: SEL(\ ) \neq \varnothing$ **and** $(\quad) \rightarrow \quad = \varnothing$
- **Consequence:** $|SEL(\ )| = 1$
- **Algorithm (start-small):**

  $\Re = \{\text{A register of label 1}\}$

  $\forall R_i \notin \Re$

  for $DATA = 0, 1$

  $\forall R_j \in \Re: WRITE_{R_j}(DATA)$ **and** $WRITE_{R_i}(\overline{DATA})$

  $\forall R_j \in \Re: READ_{R_j}$ $\mathbf{READ_{R_i}}$

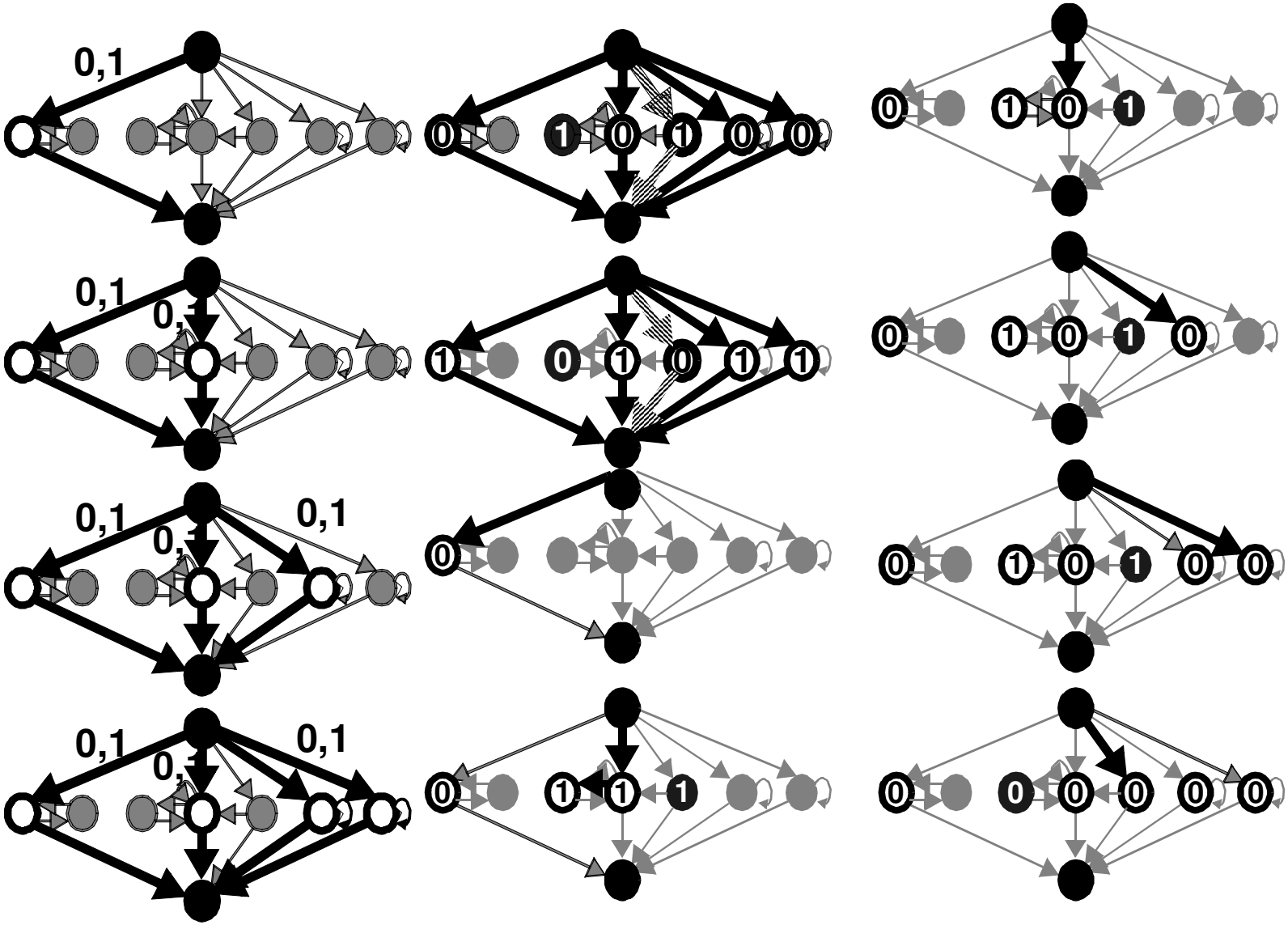
  $\Re = \Re \cup \{\mathbf{R_i}\}$

- **Proof by induction**
  - $SEL(R_i) = \varnothing \Rightarrow$ **for** $DATA = 0, 1$ **:** $\mathbf{READ_{R_i}}$ **detects the fault**
  - $\exists j: (R_j \in \Re) \wedge (SEL(R_i) \cap SEL(R_j) \neq \varnothing) \Rightarrow$ **the sequence** $WRITE_{R_j}(DATA)$ **,**

    $WRITE_{R_i}(\overline{DATA})$ **results in** $\overline{DATA}$ **in** R_j, R_i

    $\forall R_j \in \Re: (l(R_i) \geq l(R_j)) \Rightarrow R_i$ **is not used in** $READ_{R_j}$ **.**

  - **Complexity:** $O(n_R^2)$ **READ/WRITE sequences, max.** $O(n_R^3)$ **instructions**


0,1
0,1
0,1

Test of the instrucion decoding

- **Goal: detection of all single faults**
  - **≤ 1 additional activated microorder**
  - **arbitrary number of inactive microorders**
- **Method:**
  - **Mapping $R_j \rightarrow cw_j$ of registers to the code words of an error-detecting code**
  - **Faulty microorder ⇒ illegal ($DATA(\ ) \notin \{\ \}$) or wrong code $DATA(\ ) = cw_j$)**
  - **After execution of the instruction to be tested read all registers (non-destructive)**
  - **Start-small strategy:**
    - **Type 1: Single operand instructions: $f(\ ) \rightarrow R_i$ for the register set $S_1 = \{\ \}$**
      INCR, NOT, CPL, ROT
    - **Type 2: Data transfer: $R_i \rightarrow R_j$ for the set of register pairs $S_2 = \{\ \}$**
      MOV, XCHG
    - **Type 3: Data manipulation: $f(\ ) \rightarrow R_k$ for the set of register triples $S_3 = \{\ \}$**
      arithmetic, logic

Tests

- **Type 1 faults:** $\forall R_i : WRITE_{R_i}(\quad_j)$

  $\forall R_i : \forall R_j \in S_1 : \Big(\qquad . \qquad . \qquad \Big)$

- **Type 2 faults:**

  $\forall R_i : WRITE_{R_i}(\quad_j) \quad \forall R_i : \forall(\quad_j \quad_{..}) \in S_2 : \Big(\qquad . \qquad \Big)$

- **Type 2 faults:** $\forall R_i : WRITE_{R_i}(\quad_j)$

  $\forall R_i : \forall(\quad_j \quad_{..} \quad_{.}) \in S_3 : \Big(\mathrm{READ}_{R_i}, \quad ..._l, \quad ..._k, \mathrm{READ}_{R_j}$

  $READ_{R_k}, \quad ..._j, \quad ..._i, \quad ..._l \Big)$

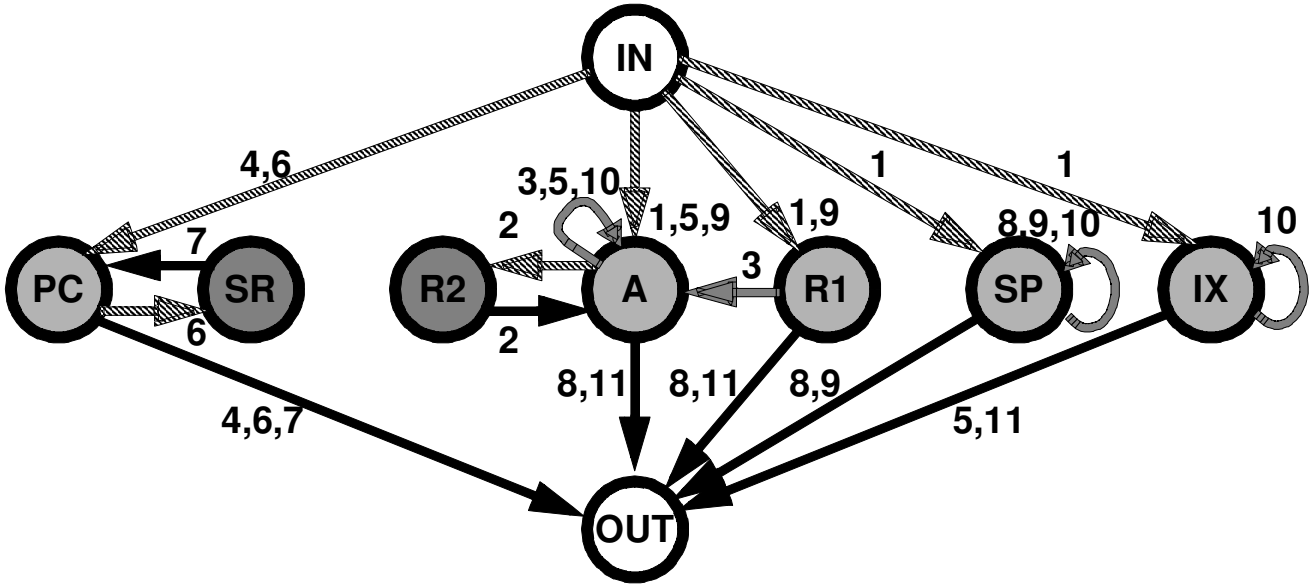
- **Example: fault during the execution of R_i :** $f(\quad_{..} \quad_j) \rightarrow R_l$
  - **Code words in R_k and R_j $\Rightarrow$ READ_{R_l} results in an illegal code**
  - **If in R_k and R_j illegal codes $\Rightarrow$ R_l potentially legal (no fault detection by the first READ_{R_l})**
    - **READ_{R_k} and READ_{R_j} potentially detects the fault**
    - **READ_{R_j} can potentially destroy the contents of R_k**
    - **after the second READ $\Rightarrow$ operands provenly correct and READ_{R_l} detects fault**
- $O(\quad_{..})$ **steps**

p-out-of-n Code with $\binom{n}{p} \geq n_R$

4-out-of-8 code	
cw_1	01101110
cw_2	10011110
cw_3	01101101
cw_4	100111101
cw_5	01101011
cw_6	10011011
cw_7	01100111
cw_8	10010111

- **for** $R_1 \leftarrow R_1 \bullet R_2$ **with** $\{ADD, AND, OR, XOR\}$
  - $ADD(R_1, R_3) \Rightarrow DATA(R_1) = 11011011 \notin \{cw_j\}$
  - $EXCHG(R_5, R_7) \Rightarrow DATA(R_5) = cw_7 \wedge DATA(R_7) = cw_5$

Example read sequence



Register	$WRITE_R$	$READ_R$	Label(R)
R2	$WRITE_A$ + *MVI A, R2*	*MVI R2, A* + $READ_A$	2
R2	$WRITE_A$ + *MVI A, R2*	PUSH *A* + *MVI R2, A* + $READ_A$ + *POP A*	2