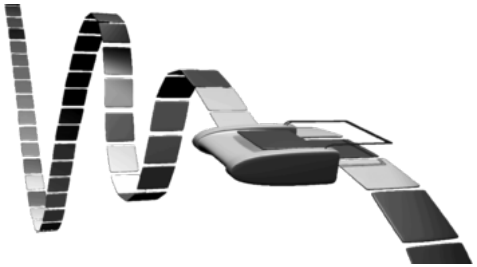
Informatika 1
2. előadás: Absztrakt számítógépek

Wettl Ferenc

Budapesti Műszaki és Gazdaságtudományi Egyetem

2015-09-08

- A Turing-gép egy $M = \langle Q, \Gamma, b, \Sigma, \delta, q_0, F \rangle$ hetes, ahol
- Q az 'állapotok' nem üres halmaza,
- Γ a 'szalag ábécé' véges, nem üres halmaza,



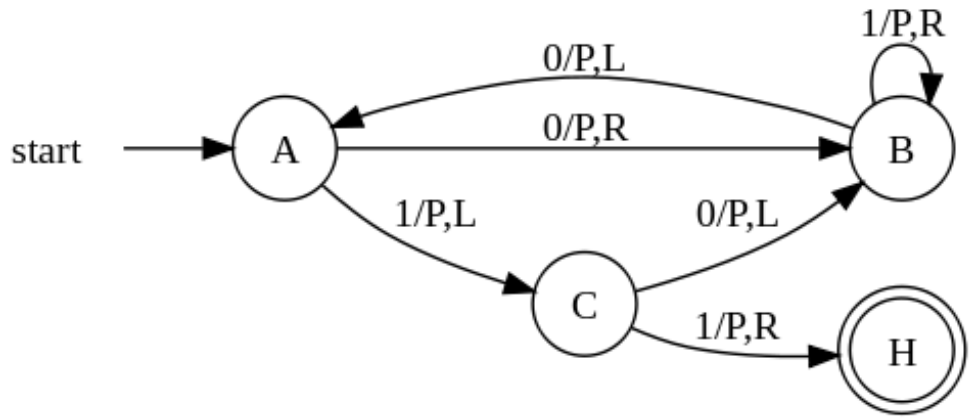
- $b \in \Gamma$ az 'üres szimbólum' (az egyetlen jel, amiből végtelen sok lehet a szalagon),
- $\Sigma \subseteq \Gamma \setminus \{b\}$ a 'bemeneti jelek' halmaza,
- $q_0 \in Q$ a 'kezdő állapot'
- $F \subseteq Q$ a 'végállapotok' halmaza (ekkor a gép leáll).
- $\delta : (Q \setminus F) \times \Gamma \hookrightarrow Q \times \Gamma \times \{L, R\}$ az 'átviteli függvény' (ha nincs értelmezve, a gép leáll), ahol R a szalag jobbra, L balra mozgatást jelenti,

H *Church–Turing-tézis*: minden formalizálható probléma, ami megoldható algoritmussal, az megoldható Turing-géppel is.

- Dolgos hód (Radó Tibor, 1962, busy beaver) az a Turing-gép, amely adott típusú Turing-gépek közül a legtöbb nem üres jelet írja egy üres szalagra, és véges lépésben leáll.

- $Q = \{A, B, C, HALT\}$
- $\Gamma = \{0, 1\}$
- $b = 0$ (az üres jel)
- $\Sigma = \{1\}$
- $q_0 = A$ (kezdő állapot)
- $F = \{HALT\}$
- δ táblázata:

	A	B	C
0	1RB	1LA	1LB
1	1LC	1RB	1RH



1	A	0	0	0	0	0	0	0	0	0	0	0	0
2	B	0	0	0	0	0	0	1	0	0	0	0	0
3	A	0	0	0	0	1	1	0	0	0	0	0	0
4	C	0	0	0	1	1	0	0	0	0	0	0	0
5	B	0	0	1	1	1	0	0	0	0	0	0	0
6	A	0	1	1	1	1	0	0	0	0	0	0	0
7	B	0	0	1	1	1	1	1	0	0	0	0	0
8	B	0	0	0	1	1	1	1	1	0	0	0	0
9	B	0	0	0	0	1	1	1	1	1	0	0	0
10	B	0	0	0	0	0	1	1	1	1	1	0	0
11	B	0	0	0	0	0	0	1	1	1	1	1	0
12	A	0	0	0	0	1	1	1	1	1	1	0	0
13	C	0	0	0	1	1	1	1	1	1	0	0	0
14	H	0	0	0	1	1	1	1	1	1	0	0	0

- Knuth: A programozás művészete c. könyve számára kigondolt hibrid (bináris/decimális) gép (assembler nyelve a MIXAL)
- Egy bájt binárisan 6 bit (0–63), decimálisan 2 számjegy (0–99). Egy szó 5 bájt és egy előjel.
- Regiszterek: A (akkumulátor) és X (extension) (5 bájt + előjel), 6 indexregiszter (2 bájt + előjel), J (jump, 2 bájt)
- Egy példaprogram:

```
TERM    EQU    19          console device no. (19=typewriter)
        ORIG   1000        start address
START   OUT    MSG(TERM)   output data at address MSG
        HLT                halt execution
MSG     ALF    "HELLO"
        ALF    " WORL"
        ALF    "D    "
        END    START       end of program
```

- Egy tutorial.

- A RAM-gép egy természetes számokkal indexelt p programtárból és egy természetes számokkal indexelt r adattárból áll, mely induláskor csak 0-kat tartalmaz.
- A program végrehajtása a p_0 cellájába írt utasítással indul és egy üres utasítással zárul.
- Az adattár i-edik cellájának ($i \in \mathbb{N}_0$) tartalmát $r[i]$ vagy r_i jelöli, ami csak egész szám lehet.
- Megengedett utasítások, ahol $z \in \mathbb{Z}$, $i, n \in \mathbb{N}_0$:

  $r_i \leftarrow z$

  $r_i \leftarrow r_n$, $r_i \leftarrow r_{r_n}$ (azaz $r_i \leftarrow r[r[n]]$),

  $r_i \leftarrow r_i \pm r_n$, ($r_i \leftarrow r_i * r_n$, $r_i \leftarrow r_i / r_n$),

  p_n: ugrás az n-edik programsorra,

  if $r_i = 0$ p_n: ugrás az n-edik programsorra, ha $r_i = 0$,

  if $r_i > 0$ p_n: ugrás az n-edik programsorra, ha $r_i > 0$,

A RAM-gép előadáson bemutatott „számítógépszerű” változata:

- A programtár és a memória véges,
- minden memóriacella 1-bytos, minden utasítás 2-bytos, az első byte az utasítást, a második az operandust tartalmazza, pl.

  `ADD 12` jelentése: $r_0 \leftarrow r_0 + r_{12}$
- számítás és összehasonlítás csak a 0-dik memóriacella (és esetleg egy másik) tartalmával végezhető,
- az utasításokat mnemonikokkal jelöljük, ezek három változata:
  - közvetlen: az n operandus egy szám, a műveletet azzal végezzük (jelölése az utasítás végére tett =)
  - direkt: az n operandust memóriacímnek tekintjük és a műveletet az $r[n]$ (azaz az r_n) tartalmával végezzük,
  - indirekt: az n operandust egy memóriacím címének tekintjük, a műveletet az $r[r[n]]$ (azaz az r_{r_n}) számmal végezzük (jelölése az utasítás végére tett *)

Vezérlő utasítások

`JUMP`	n	ugrás az n-edik utasításra
`JZERO`	n	ugrás az n-edik utasításra, ha $r_0 = 0$
`JGTZ`	n	ugrás az n-edik utasításra, ha $r_0 > 0$
`HALT`		leállás

Aritmetikai utasítások

	direkt		*indirekt*		*közvetlen op*	
`ADD` n	$r_0 \leftarrow r_0 + r_n$	`ADD*` n	$r_0 \leftarrow r_0 + r_{r_n}$	`ADD=` n	$r_0 \leftarrow r_0 + n$
`SUB` n	$r_0 \leftarrow r_0 - r_n$	`SUB*` n	$r_0 \leftarrow r_0 - r_{r_n}$	`SUB=` n	$r_0 \leftarrow r_0 - n$
`MULT` n	$r_0 \leftarrow r_0 * r_n$	`MULT*` n	$r_0 \leftarrow r_0 * r_{r_n}$	`MULT=` n	$r_0 \leftarrow r_0 * n$
`DIV` n	$r_0 \leftarrow r_0 / r_n$	`DIV*` n	$r_0 \leftarrow r_0 / r_{r_n}$	`DIV=` n	$r_0 \leftarrow r_0 / n$

Adatmozgatás, IO

	direkt		*indirekt*		*közvetlen op*	
`LOAD` n	$r_0 \leftarrow r_n$	`LOAD*` n	$r_0 \leftarrow r_{r_n}$	`LOAD=` n	$r_0 \leftarrow n$
`STORE` n	$r_n \leftarrow r_0$	`STORE*` n	$r_{r_n} \leftarrow r_0$		
`READ` n	az input eszközről $r_1, r_2, \ldots, r_n$-be olvas n számot				
`WRITE` n	az output eszközre írja $r_1, r_2, \ldots, r_n$ tartalmát				

Írjunk programot (a, b) kiszámítására, ahol $a, b \in \mathbb{N}_0$!

```
 p   utasítás operandus  megjegyzés
 0   READ       2        beolvas két számot (r[1] <- a és r[2] <- b)
 1   LOAD       2        r[0] <- r[2] azaz r[0] <- b
 2   JZERO     16        ha b = 0, ugorj 16-ra, a lnko(a,b) = a
 3   LOAD       1        r[0] <- r[1]
 4   DIV        2        r[0] <- ⌊a/b⌋
 5   STORE      3        r[3] <- ⌊a/b⌋
 6   MULT       2
 7   STORE      4        r[4] <- b*⌊a/b⌋
 8   LOAD       1
 9   SUB        4        r[0] <- a - b*⌊a/b⌋ = a mod b
10   STORE      5
11   LOAD       2
12   STORE      1        r[1] <- b
13   LOAD       5
14   STORE      2        r[2] <- a mod b
15   JUMP       2
16   WRITE      1        írd ki r[1] tartalmát, ami = lnko(a,b)
17   HALT
```

Kérdések

1. Hogy működik a Turing-gép?
2. Melyik Turing-gépre írt programokat nevezzük dolgos hódnak?
3. A RAM-gép Neumann-elvű?
4. Mi a különbség a direkt és az indirekt gépi utasítás között?
5. Mi a memória tartalma az alábbi program végrehajtása után?

```
1    LOAD=     5
2    STORE     1
3    STORE*    1
4    JZERO     7
5    LOAD=     2
6    MUL       1
7    HALT
```