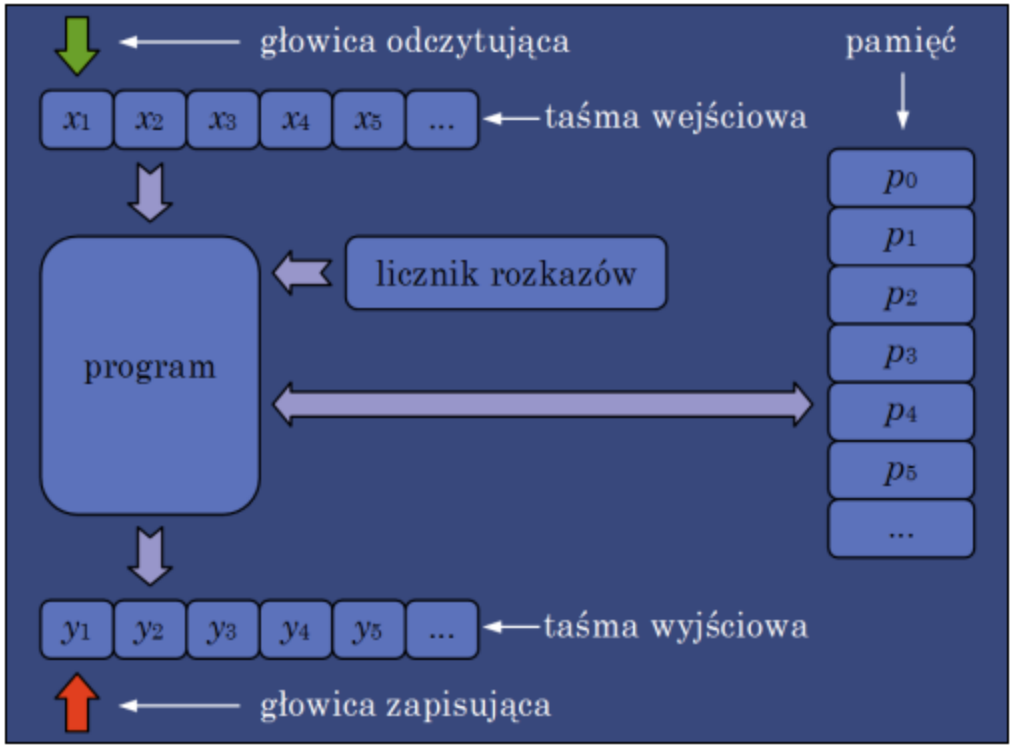
SDIZO-MASZYNA RAM



Schemat budowy Maszyny RAM

SDIZO-MASZYNA

Typy operandów:
=n – liczba
n – adresowanie bezp.
^n – adresowanie pośr.

rozkaz	argument	działanie
LOAD	n	M(0) := M(n)
	=n	M(0) := n
	^n	M(0) := M(M(n))
STORE	n	M(n) := M(0)
	^n	M(M(n)) := M(0)
READ	n	M(n) := kolejna liczba odczytana z taśmy wejściowej
	^n	M(M(n)) := kolejna liczba odczytana z taśmy wejściowej
WRITE	n	wypisanie na taśmie wyjściowej wartości M(n)
	^n	wypisanie na taśmie wyjściowej wartości M(M(n))
	=n	wypisanie na taśmie wyjściowej wartości n
ADD	n	M(0) := M(0) + M(n)
	^n	M(0) := M(0) + M(M(n))
	=n	M(0) := M(0) + n
SUB	n	M(0) := M(0) - M(n)
	^n	M(0) := M(0) - M(M(n))
	=n	M(0) := M(0) - n
MULT	n	M(0) := M(0) * M(n)
	^n	M(0) := M(0) * M(M(n))
	=n	M(0) := M(0) * n
DIV	n	M(0) := M(0) / M(n)
	^n	M(0) := M(0) / M(M(n))
	=n	M(0) := M(0) / n
MOD	n	M(0) := M(0) mod M(n)
	^n	M(0) := M(0) mod M(M(n))
	=n	M(0) := M(0) mod n
JUMP	etykieta	przejście do wykonania instrukcji, która jest poprzedzona etykietą etykieta
JZERO	etykieta	przejście do wykonania instrukcji, która jest poprzedzona etykietą etykieta, pod warunkiem że M(0)=0
JGTZ	etykieta	przejście do wykonania instrukcji, która jest poprzedzona etykietą etykieta, pod warunkiem że M(0)>0
JLTZ	etykieta	przejście do wykonania instrukcji, która jest poprzedzona etykietą etykieta, pod warunkiem że M(0)<0
HALT		zatrzymanie wykonywania programu

SDIZO-MASZYNA RAM

Kryterium kosztu jednorodnego mówi, że każda instrukcja kodu RAM wymaga jednej jednostki czasu, natomiast każda komórka pamięci, jednej jednostki pamięci.
Kryterium kosztu logarytmicznego uwzględnia długość operandu (co jest bardziej miarodajne dla programów operujących na liczbach o dowolnej wielkości). Innymi słowy - koszt wykonania instrukcji jest proporcjonalny do długości operandów tych instrukcji.
Zdefiniujmy funkcję lcost(x): *(nazwa pochodzi od logarithmic cost)*
lcost(x) = $\lceil \log | x | \rceil$ dla $x \neq 0$, oraz
lcost(x) = 1 dla x = 0.
lcost(x) = 0 dla x o nieokreślonej wartości
Dla krótszego zapisu zdefiniujemy sobie t(a) dla trzech możliwych postaci operandu a.

Operand a Koszt t(a)
=x => lcost(x)
x => lcost(x) + lcost(m(x))
^x => lcost(x) + lcost(m(x)) + lcost(m (m(x)))

SDIZO-MASZYNA RAM

Kryterium kosztu logarytmicznego opiera się na założeniu, że koszt wykonania instrukcji jest proporcjonalny do długości operandów tych instrukcji. Aby lepiej zobrazować obliczanie kosztu, posłużmy się przykładem i policzmy koszt dla instrukcji MULT ^x. Najpierw musimy ustalić koszt, gdzie t(a) jest kosztem operandu a. Aby rozpoznać liczbę całkowitą x przez maszynę, potrzeba czasu lcost(x). Następnie, aby odczytać m(x) (zawartość komórki o indeksie x) oraz odszukać rejestr m(x) potrzeba czasu lcost(m(x)). Z kolei czytanie zawartości rejestru m(x) kosztuje lcost(m(m(x))). Skoro instrukcja MULT ^x mnoży liczbę całkowitą m(m(x)) przez m(0), widzimy, e ostatecznym kosztem, jaki należy przypisać instrukcji MULT ^x, jest lcost(m(0)) + lcost(x) + lcost(m(x)) + lcost(m(m(x))). Logarytmiczną złożoność pamięciową programu RAM definiujemy jako sumę lcost(ix) po wszystkich komórkach pamięci, gdzie ix jest największą liczbą całkowitą, jaka była umieszczona w komórce pamięci o indeksie x podczas obliczeń. Przy obliczaniu sumy nie będzie nam przeszkadzało, że większość komórek będzie zawierała wartość nieokreśloną (czyli taką, jaką posiadają przed uruchomieniem programu wszystkie komórki pamięci), ponieważ dla takiego przypadku koszt wynosi 0 (patrz: definicja funkcji lcost). Z powyższego jasno wynika, że dany program może mieć całkowicie różne złożoności czasowe i pamięciowe zależnie od tego, czy użyje się do szacowania kosztu jednorodnego czy logarytmicznego. Jeżeli zakładamy, że każda liczba zajmuje jedną jednostkę pamięci lub po prostu: stałą liczbę jednostek, wówczas stosujemy koszt jednorodny (np.: analizując program napisany w języku C, używający zmiennych typu int). W przeciwnym razie, dla realistycznej analizy złożoności, bardziej właściwi może być koszt logarytmiczny (w szczególności – analizując programy w kodzie RAM).

SDIZO-MASZYNA RAM

Instrukcja	Koszt
LOAD *a*	$t(a)$
STORE *x*	$lcost(m(0)) + lcost(x)$
STORE ^*x*	$lcost(m(0)) + lcost(x) + lcost(m(x))$
ADD *a*	$lcost(m(0)) + t(a)$
SUB *a*	$lcost(m(0)) + t(a)$
MULT *a*	$lcost(m(0)) + t(a)$
DIV *a*	$lcost(m(0)) + t(a)$
READ *x*	$lcost(\text{input}) + lcost(x)$
READ ^*x*	$lcost(\text{input}) + lcost(x) + lcost(m(x))$
WRITE *a*	$t(a)$
JUMP *e*	1
JGTZ *e*	$lcost(m(0))$
JZERO *e*	$lcost(m(0))$
HALT	1

t(a) – koszt operandu a (zależy od sposobu adresowania)

http://mmsyslo.pl/ram/maszyna.html
http://www.szkup.com/download/MaszynaRAM.pdf