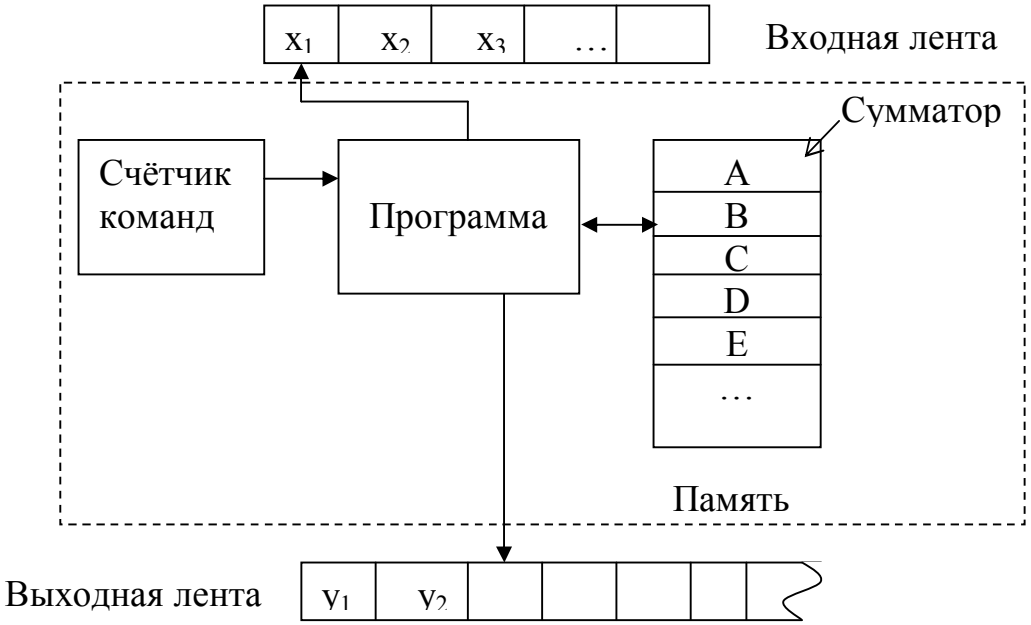
Машины с произвольным доступом к памяти (PAM)

Машина с произвольным доступом к памяти (или, иначе равнодоступная адресная машина - PAM) моделирует вычислительную машину с одним сумматором, в которой команда программы не могут изменять сами себя.

PAM состоит из входной ленты, с которой она может только считывать, и выходной ленты, на которую можно записывать, и памяти.



Входная лента разделена на ячейки, каждая из которых может содержать число (десятичное, двоичное, возможно отрицательное). Всякий раз, когда символ считывается с входной ленты, читающая головка сдвигается на одну ячейку вправо.

Выходная лента также разбита на ячейки, которые пусты перед началом работы. При выполнении команды записи в той ячейке выходной ленты, которую в текущий момент обозревает головка, печатается целое

число, и головка затем сдвигается на одну ячейку вправо. Как только *выходной символ записан, его уже нельзя изменить.*

Память состоит из последовательности регистров a, b, c, d …, каждый из которых способен хранить произвольное целое число.

Все вычисления производятся в первом регистре A, который называется Сумматором.

На число регистров верхняя граница не устанавливается. Такая идеализация допустима, если:

1. Размер задачи мал.
2. Целые числа, участвующие в вычислениях, достаточно малы, чтобы их можно было помещать в одну ячейку.

Программа для РАМ (или РАМ-программа) **не записывается в память**. Поэтому мы предполагаем, что программа не изменяет сама себя.

Программа представляет последовательность команд, помеченных некоторым образом. Эти команды напоминают те, которые встречаются в реальных вычислительных машинах.

Мы предполагаем, что имеются арифметические команды, команды ввода-вывода, косвенная адресация (например, для индексации массивов) и команды разветвления.

Рассмотрим пример набора команд. Каждая команда состоит из двух частей: кода операции и адреса, т.е. номера той ячейки памяти, которая отводится для хранения значения операнда (операнд - величина, участвующая в операции).

В качестве операнда могут быть использованы следующие значения:

1. =i - означает само целое число i и называется литералом
2. i - адрес операнда, содержимое регистра i (i≥0)
3. *i - косвенная адресация, т.е. значением операнда служит содержимое регистра j, где j - целое число, находящееся в регистре i, если j<0 машина останавливается.

Значение программы P можно определить с помощью двух объектов:

1. *отображения C*: задаёт отображение множества неотрицательных чисел (т.е. адресов регистров) во множество целых чисел (т.е. содержимое этих регистров)
2. *счётчик команд*, который определяет очередную выполняемую команду.

Отображение C задаёт функцию C(i), её значение - целое число, содержащееся в регистре i (содержимое регистра i).

В начале работы C(i) = 0, для любого i≥0 (т.е. содержимое всех регистров равно 0, пусто)

Счётчик команд установлен на первую команду программы P, а выходная лента пуста. После выполнения к-ой команды программы P счётчик команд автоматически переходит на к + 1 команду (т.е. на очередную команду), если только предыдущая команда не была командой условного, безусловного перехода или останова.

Для описания действий команд зададим значение V(a) операнда a:

V(=i) = i - само значение операнда i

V(i) = C(i) - адрес - целое число, содержащееся в регистре i

V(*i) = C(C(i)) - содержимое регистра C(i)

Таблица действий команд РАМ-машины.

	Код операции	Действие	Команда	Описание действия
1.	LOAD	загрузка (вызов в сумматор)	L a	C(A) ← V(a) в сумматор A загружается a
2.	STORE	поместить в память	ST i	C(i) ← C(A) в регистр с адресом i поместить содержимое сумматора
			ST *i	C(C(i)) ← C(A) в регистр с адресом C(i) поместить содержимое сумматора
3.	ADD	сложение	A a	C(A) ← C(A) + V(a)
4.	SUB	вычитание	S a	C(A) ← C(A) - V(a)
5.	MULT	умножение	M a	C(A) ← C(A) * V(a)
6.	DIV	деление	D a	C(A) ← [C(A) / V(a)] целая часть от деления
7.	READ	ввод	R i	C(i) ← очередной входной символ
			R *i	C(C(i)) ← очередной входной символ
8.	WRITE	вывод	W a	V(a) значение операнда a печатается в обозреваемой ячейки выходной ленты, затем головка сдвигается на одну ячейку вправо.
9.	JUMP	безусловный переход	J b	Счётчик команд устанавливается на команду с меткой b
10	JGTZ	уловный переход	JG b	Если C(A) > 0, то счётчик команд

				устанавливается на команду с меткой b, в противном случае на следующую команду
11	JZERO	условный переход при равенстве 0	JZ b	Если C(A) = 0, то счётчик команд устанавливается на команду с меткой b, в противном случае на следующую команду
12	HALT	останов	H	Работа программы прекращается

В принципе, можно было бы к нашему набору добавить любые другие команды, встречающиеся в реальных вычислительных машинах, такие, как логические или символьные операции, и при этом, порядок сложности задач не изменится.

РАМ-программа задаёт отображение множества символов, записанное на входной ленте, во множество выходных символов, т.е.

$P : RX \rightarrow RY$

Так как при некоторых входных данных программа P может не останавливаться, то это отображение является частичным (т.е. для некоторых входов оно может быть не определено).

Это отображение может быть использовано для преобразования числовых и символьных данных.

В первом случае его можно интерпретировать как функцию, заданную на множестве входных числовых данных, и имеющую значения на множестве числовых данных, которые записываются на ленту, т.е.

$y = f(x_1, x_2, \ldots, x_n)$

А во втором случае программа P является распознавателем некоторого языка.

Языком, допускаемым программой P, называется множество всех входных цепочек (слов), которые воспринимаются и обрабатываются программой.

Пример:

Какие действия выполняют команды **LOAD a**, если a принимает значения равны i, i и *i, а также известно, что i=5, C(5)=7, C(7)=12.

1. LOAD =i	LOAD =5	C(A) ← 5
2. LOAD i	LOAD 5	C(A) ← C(5) = 7
3. LOAD *i	LOAD *5	C(A) ← C(C(5))=C(7) = 12

Вычислительная сложность PAM-программ.

Эффективность алгоритмов характеризуют временная и емкостная сложности, которые рассматриваются как функции размера входа n. Мы будем рассматривать сложность в худшем случае и среднюю сложность.

Если при данном размере входа n в качестве меры сложности берётся максимальная из сложностей (по всем входам этого размера), то она называется **сложностью в худшем случае**.

Если в качестве меры сложности берётся "средняя" сложность по всем входам данного размера, то она называется **средней** (или **усреднённой**) **сложностью**.

Временная сложность в худшем случае (или просто временная сложность) **РАМ-программы** - это функция T(n), равная наибольшей (по всем входам размера n) из сумм времён затраченных на каждую выполненную команду.

Временная сложность в среднем - это среднее значение, взятое по всем входам размера n, тех же самых сумм.

Емкостная сложность в худшем случае РАМ-программы - это функция S(n), равная наибольшей (по всем входам размера n) из сумм емкостей всех регистров, к которым было обращение.

Емкостная сложность в среднем - это среднее значение сумм.

Чтобы точно определить временную и емкостную сложность, надо указать время, необходимое для выполнения каждой РАМ-команды, и объём памяти, используемый каждым регистром.

Рассмотрим два весовых критерия:

- **Равномерный весовой критерий**. При равномерном весовом критерии каждая РАМ-команда затрачивает одну единицу времени, и каждый регистр использует одну единицу памяти.
- **Логарифмический весовой критерий**. Он более реалистичен и учитывает ограниченность размера реальной ячейки памяти.

Пусть l(i) - логарифмическая функция на целых числах, определим её следующим образом:

$$l(i)=\begin{cases}[\log|i|+1, & если\ i\neq 0\\ 1 & ,если\ i=0\end{cases},$$

тогда логарифмические веса t(a) для всех трёх возможных типов операнда a имеют вид:

Операнд a	Вес t(a)
=i	l(i)
i	l(i) + l(C(i))
*i	l(i) + l(C(i)) + l(C(C(i)))

Рассмотрим веса РАМ-команд.

	Команда	Вес
1.	L a	t(a)
2.	ST i	l(C(A)) + l(i) + l(C(i))
	ST *i	l(C(A)) + l(i) + l (C(i)) + l(C(C(i)))
3.	A a	l(C(A)) + t(a)
4.	S a	l(C(A)) + t(a)
5.	M a	l(C(A)) + t(a)
6.	D a	l(C(A)) + t(a)
7.	R i	l(вход) + l(i) + l(C(i))
	R *i	l(вход) + l(i) + l (C(i)) + l(C(C(i)))
8.	W a	t(a)
9.	J b	1
10.	JG b	l(C(A))
11.	JZ b	l(C(A))
12.	H	1

Здесь учитывается, что для представления целого числа n в регистре требуется [log n] + 1 битов (бит - единица памяти). Регистры же (по нашему допущению) могут содержать произвольно большие числа.

Логарифмический весовой критерий основан на грубом допущении, что **вес каждой команды** (или цена выполнения команды) **пропорционален длине её операндов**.

Пример:

Рассмотрим вес команды **ADD a**, если a=*i

ADD *i C(A)← C(A) + C(C(i))

Временная сложность команды определяется следующим образом: просмотр целого числа i занимает время l(i), затем, чтобы прочитать содержимое C(i) регистра i и определить его местоположение, понадобиться время l(C(i)), наконец считывание содержимого из регистра C(i) требует время l(C(C(i))).

Так как мы все действия выполняем в сумматоре, складывая найденое число с содержимым сумматора, то для этого нужно время l(C(A)). Следовательно вес ADD *a:

$l(C(A)) + l(i) + l(C(i)) + l(C(C(i)))$

определим логарифмическую сложность РАМ-программы как сумму по всем работавшим регистрам, включая сумматор, величин l(i), где x_i - наибольшее по абсолютной величине целое число, содержащееся в i-ом регистре за время вычислений.

Пример:

Оценим временную и емкостную сложность РАМ-программы, вычисляющей функцию $f = n^n$.

При подсчёте временной сложности будет доминировать цикл с командой MULT, которая выполняется (n-1) раз.

Когда команда выполняется i-ый раз сумматор A содержит n^i, а регистр C содержит n. При равномерном весовом критерии (как мы отмкчали) на выполнение каждой команды MULT потребуется 1 единица времени, и поэтому на выполнение всех команд умножения тратится время O(n).

При логарифмическом весовом критерии i-я команда умножения занимает время

MULT a $l(C(A)) + t(a)$

i- MULT $l(n^i) + l(n) \approx [\log |n^i|] + 1 + \log n \approx (i + 1) \log n$

Тогда время выполнения всех команд умножения равно:

$\sum_{i=1}^{n-1} (i+1) \log n$, что составляет $O(n^2 \log n)$

Емкостная сложность определяется числами, которые хранятся в регистрах от A до D. При равномерном весовом критерии емкостная сложность составляет O(1), а при логарифмическом - O(), т.к. наибольшее целое число среди содержащихся в этих регистрах есть n^n , а :

$\log (n^n) = [\log |n^n|] + 1 \approx n \log n$

Таким образом, сложности функции n^n таковы:

	Равномерный вес	Логарифмический вес
Временная сложность	O(n)	$O(n^2 \log n)$
Емкостная сложность	O(1)	O(n log n)

Рассмотрим целесообразность оценки при использовании равномерного и логарифмического веса.

Для этой программы равномерный вес реалистичен только в ситуации, когда столь большое целое, как n^n , записывается в виде одного машинного слова.

Если же значение n^n превышает одно машинное слово, то тогда даже логарифмическая временная сложность до некоторой степени нереалистична, т.к. она предполагает, что 2 целых числа i и j перемножаются за время O(l(i) + l(j)), а возможность этого неизвестна.