Аннотация

В настоящем документе содержатся предпосылки создания и основные концепции параллельной виртуальной машины, а также пояснения к реализации VPM (версия 1.0.1). Документ содержит также спецификацию команд и формата входной программы машины. Он является рекомендуемым стандартом для создания собственно параллельной виртуальной машины и комплекса специального программного обеспечения для подготовки ПО для данной машины (ассемблера и компилятора с языка OCCAM).

Содержание

Введение

В настоящее время приоритетным направлением развития вычислительной техники признано построение параллельных вычислительных систем. Это обеспечивает возможность эффективного решения целого класса математических задач за счёт одновременного выполнения отдельных составляющих программы.

Существуют два основных подхода к построению систем параллельных вычислений:

- Централизация – выделение явного управляющего центра, предназначенного для решения задач распределения ресурсов, потоков управления. Достоинства данного подхода: применимость стандартных алгоритмов, увеличение количества доступных ресурсов с увеличением числа процессоров. Недостатки подхода: сложность распараллеливания кода, проблемы разделяемых ресурсов, «эффект насыщения» - падение производительности при увеличении количества процессоров свыше некоторого порогового значения.
- Децентрализация – отсутствие явного управляющего центра при равенстве приоритетов процессоров. Достоинства подхода: простота выполнения специализированного кода, эффективность при реализации парадигмы управления потоком данных, удобство построения больших систем, отсутствие эффекта насыщения. Недостатки: иные способы программирования.

Для проверки эффективности второго подхода создана виртуальная параллельная машина (VPM) реализующая децентрализованный подход. Основными принципами построения являются: полная децентрализация управления, все процессоры равноправны, отсутствуют разделяемые ресурсы (общая память, глобальные переменные). Программа представляет собой набор

процессов, выполняющихся независимо. Процессы взаимодействуют посредством небуферизуемых каналов. Машина представляет собой набор программных модулей, реализующих виртуальные процессоры, и функционирует на базе компьютерной сети.

Программирование для VPM состоит из построения решётки процессов, где на входной канал корня подаётся поток данных, а из выходного потока терминального узла читается результирующий поток данных, и описания поведения всех процессов. В качестве базового формализма для программирования выбран формализм машины с неограниченными регистрами (МНР) с добавлением операций работы с каналами. Языком высокого уровня, реализующего расширенный вариант данного формализма, является модифицированный язык Оккам. Исполнимый код для виртуальной машины транслируется с этого языка специально разработанным компилятором.

Основные концепции

1.1. Регистр – единица хранения информации. Не может содержать агрегированных данных. Все регистры содержат данные одного типа.

1.2. Команда (инструкция) – единица обработки данных. Минимально необходимый набор – набор команд формализма МНР.

1.3. Процесс – последовательность инструкций, выполняемых последовательно. Процессы выполняются независимо друг от друга. Каждый процесс обладает своим набором регистров.

1.4. Канал – средство обмена информацией между процессами. Поддерживает два вида команд – записать значение в канал и прочитать из канала и поместить в регистр. Канал не буферизуется, т.е. обмен по каналу возможен только при готовности обоих взаимодействующих процессов, иначе выполнение операции приостанавливается до готовности второго взаимодействующего процесса.

Описание реализации.

1.5. Программа на Оккаме при предварительной трансляции разбивается на набор процессов, которые запускаются одновременно. Содержимое блоков PAR представляется как набор процессов, ожидающих чтения из канала.

1.6. Каждый процесс представляет собой отдельный интерпретатор с кодом исполняемой программы.

1.7. Машина по структуре аналогична сети транспьютеров. Топология сети известна перед началом работы. Все выделенные процессы на этапе запуска распределяются по виртуальным процессорам. Если процессоров (машин в компьютерной сети) меньше, чем процессов, то дополнительные процессы запускаются на одной физической машине.

1.8. Реализация каналов производится посредством менеджера каналов – отдельного программного блока, скрывающего от интерпретаторов специфику канала – локальный/сетевой.

Набор команд машины.

Набор команд MHP – минимален, необходим, но неэффективен, поэтому вводится следующий набор классов команд:

1.9. Манипуляция с регистрами: загрузка непосредственного значения и пересылка регистр – регистр.

1.10. Регистровая арифметика (возможно стековая – из-за простоты компиляции). Включает логические операции.

1.11. Операции с каналами. Каналы нумеруются.

1.12. Команды переходов, условных и безусловных.

1.13. Команды управления процессом.

1.14. Дополнительные команды (ввод/вывод на экран/с клавиатуры/из файла).

VPM имеет гарвардскую архитектуру (память программ и память данных разделены)

Система команд.

Программа представляет собой последовательность команд с расположенными за ними операндами. Байты операндов хранятся в обратном порядке.
Каждая команда представлена 2+k байтами.

1 байт (группа команды)

Группа команды				Количество аргументов ()			

2 байт (номер команды)

Номер команды в группе							

2+k байт (тип операнда k)

Размер операнда			Резерв (равны 0)			Тип операнда	
			0	0	0		

Тип операнда

Номер	Биты	Описание
0	0000	R – регистр
1	0001	V – непосредственное значение
2	0010	I – регистр с номером в регистре
3	0011	C – канал

Размер операнда

Номер	Биты	Описание
0	000	Нет операнда
1	001	1 байт
2	010	2 байт
3	011	4 байт
4	100	8 байт (float)
5..7		Резерв (не должны использоваться)

*) элементы, помеченные цветом, не рекомендуются использовать в данной версии (т.е. не гарантируется корректное исполнение)

Если тип операнда R, I то размер операнда указан для регистра
Адресация регистров и каналов ведется всегда 32-битными словами

Группа команды

Номер	Биты	Описание
0	0000	Управляющие
1	0001	Переходы
2	0010	Регистровые
3	0011	Битовые
4	0100	Логические
5	0101	Математические
6..15		Резерв (не должны использоваться)

Номера команд

Второй байт	**Группа команд**					
	0	1	2	3	4	5
0	NOP	JMP	MOV	SETZ	AND	ADD

1	RET	JZ		RESZ	OR	SUB
2	START	JNZ		SETC	XOR	MUL
3	KILL	JC		RESC	NEG	DIV
4		JNC				MOD

Действие команд на флаги

Если команда имеет операнд результата и после выполнения команды результат равен 0, то Z=1

Если команда имеет операнд результата и после выполнения команды результат не равен 0, то Z=0

Если команда имеет операнд результата и после выполнения команды результат при преобразовании типа не подлежит усечению, то C=1

Если команда имеет операнд результата и после выполнения команды результат не подлежит усечению, то C=0

Состояние регистров перед запуском процесса

IP	C	Z				
0	0	0				

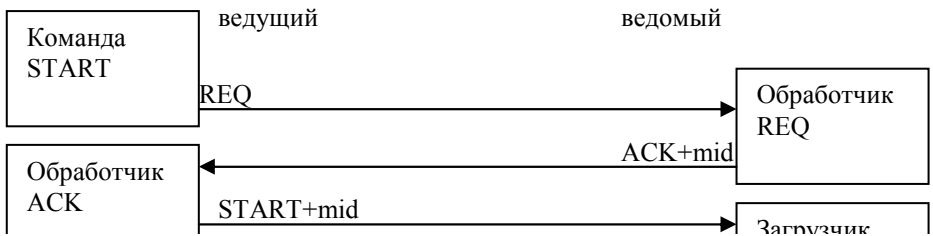
Стандартные каналы

Канал 0 –стандартный ввод

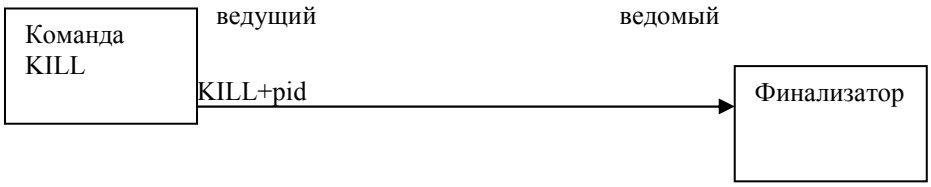
Канал 1 –стандартный вывод

Спецификация взаимодействий между вычислителями

Порождение процесса



Уничтожение процесса



Примеры канальных взаимодействий процессов

Запросы процесса 1	Действие менеджера	Состояние виртуального канала			Действие менеджера	Запросы процесса 2
		R	DATA	W		
…						…
Чтение	→REQ	+				
Чтение	Ожидание	+				
		+	+	+	←DATA	Запись
	←DATA				→ACK	
…						…
Запись	→ DATA		+	+		
		+	+	+	← REQ	Чтение
	←ACK				→DATA	

Примечание: в таблице приводится логика взаимодействия процессов на модели канала без учета особенностей его реализации.

Спецификация файла задания.

Задание представляет собой совокупность файлов исполнимого кода процессов и файла задания. Файл задания имеет имя task.ini и имеет следующую структуру:

```
[main]
Description=Смысловое наименование задания (строка win1251)
ProcCount=количество процесоов
StartProc=номер процесса, запускаемого первым
ChanCount=требуемое число каналов

[proc1]
MemorySize=количество памяти для процесса 1
FileName=имя файла исполнимого кода процесса 1

[proc2]
MemorySize=количество памяти для процесса 2
FileName=имя файла исполнимого кода процесса 2

…….

[procK]
MemorySize=количество памяти для процесса K
FileName=имя файла исполнимого кода процесса K
```

Файл исполнимого кода процесса представляет собой непосредственную последовательность машинных команд.

Работа машины.

При работе машины в текущей директории должен находиться файл network.ini с параметрами сети. Файл network.ini имеет следующую структуру:

[Main]
Host= диапазон IP для запросов вычислителей
Port= рабочий порт

Работа может быть инициирована на любом вычислителе путем запуска проекта. Все нештатные ситуации вычислителей отображаются в окне состояния.

Входным каналом чтения (канал 0) по умолчанию для корневого вычислителя является файл input.

Входным каналом записи (канал 1) по умолчанию для корневого вычислителя является файл output.

Дополнительные возможности при работе:

- Просмотр состояния выполнения (dump) с отображением фрагмента памяти. Dump происходит также при исключениях в работе машины.
- Принудительное завершение выполнения пользователем.
- Отображение эффективности вычисления единичного вычислителя (в mips)

Пример задания для VPM.

Задание для вычисления факториала для числа

Файл задания:
[main]

```
Description=Factorial calculation
ProcCount=2
StartProc=2
ChanCount=100

[proc1]
MemorySize=4096
FileName=1.prc

[proc2]
MemorySize=4096
FileName=2.prc
```

Процесс 1:

03 02 61 61 61 02 00 00 00 02 00 00 00 03 00 00 00	start 2,2,3
03 02 61 61 61 02 00 00 00 04 00 00 00 05 00 00 00	start 2,4,5
22 00 63 61 02 00 00 00 00 00 00 00	mov (2),0
22 00 63 61 02 00 00 00 05 00 00 00	mov (2),5
22 00 63 61 04 00 00 00 05 00 00 00	mov (4),5
22 00 63 61 04 00 00 00 0A 00 00 00	mov (4),0A
22 00 60 63 00 00 00 00 03 00 00 00	mov [0],(3)
22 00 60 63 04 00 00 00 05 00 00 00	mov [4],(5)
52 02 60 60 00 00 00 00 04 00 00 00	mov [0],(4)
22 00 63 60 01 00 00 00 00 00 00 00	mov (1),[0]
00 01	ret

Процесс 2:

22 00 60 61 00 00 00 00 01 00 00 00	mov [0],1
22 00 60 63 04 00 00 00 00 00 00 00	mov [4],(0)
22 00 60 63 0C 00 00 00 00 00 00 00	mov [c],(0)
52 00 60 61 04 00 00 00 01 00 00 00	add [4],1
52 02 60 60 00 00 00 00 04 00 00 00	mul [0],[4]
22 00 60 60 08 00 00 00 04 00 00 00	mov [8],[4]

52 01 60 60 08 00 00 00 0C 00 00 00	sub [8],[c]
11 02 61 24 00 00 00	jnz loop
22 00 63 60 01 00 00 00 00 00 00 00	mov (1),[0]
01 00	ret

Примечание: [] – регистровая адресация
() - канальная адресация
{} – косвенная регистровая адресация

Словарь терминов для VPM.

Вычислитель – единичная виртуальная машина, которая доступна в пространстве решений и может выполнять не более одного процесса. Все множество вычислителей образует пространство решений.

Задание – совокупность исполнимых кодов процессов и файла описания задания для решения конкретной задачи на VPM.

Канал – средство обмена данными и синхронизации между процессами. Вся совокупность каналов образует пространство каналов.

Команда – минимальная единица описания действия вычислителя.

Память данных - вся совокупность регистров, доступных некоторому процессу и предоставляемая соответствующим вычислителем.

Процесс – последовательность команд для одного вычислителя.

Регистр – минимально возможная адресуемая единица хранения информации в памяти данных.

