情報科学特別講義A

for文における自作VMと C++における実装の実行時間比較

第二回発表資料

発表者 阿部 碧音

目次

変更点

・オペランドの個数の無制限化
　→ 引数の変更による命令の汎用化が可能に

・ラベル機能をMap実装に変更
　→ C++の Map の find を利用した実装を採用。
　　find は Map のサイズに対して対数時間で動作するため
　　従来の vector を線形で探索するよりも高速に。

・ GT, LT (大小判定), LOGNOT(論理否定) の追加

・Store, Load 命令をGetLocal, SetLocal へと変更
　→ スコープの概念に対応、デバッグが不十分の可能性がある。

試験内容

A = x * i + y; の演算を測定する。
x = 10 (const), y = 100 (const), i はfor文のインデックス値。

for文の繰り返し回数は (100, 1000, …, 10^8) で実行。

時間は Windows Powershell の Measure-Command を利用して測定。

出力時間を含めたくなかったため、測定時に出力をし答え合わせを行うということはしなかった。
答えはあらかじめ合っていることを確認している。

実行環境

ハードウェア

OS	Windows11 Home 22H2
CPU	Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz
RAM	DDR4 32GB
ストレージ	Hanye ME70-2TA01 PCIe M.2 2TB

C++コンパイラ

コンパイラ	GCC-8.2.0
オプション	指定なし

試験内容

C++での実装

```cpp
#include <iostream>

int main(void)
{
    for (int i = 0; i < 100; i++) {
      int A = 10 * i + 100;
    }

    return 0;
}
```

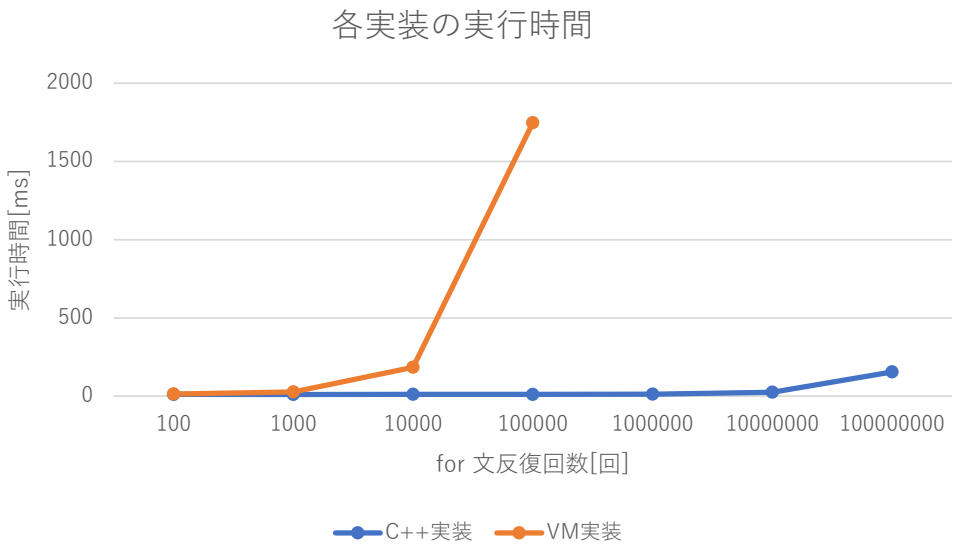
自作VMでの実装

```
PUSH 0                  // イテレータ
SETLOCAL i
POP
PUSH 1000000            // 反復終了回数
SETLOCAL ForEndNum
POP
BeginForLabel:          // for開始
PUSH 10
GETLOCAL i
MUL                      // 10 * i
PUSH 100
ADD                      // + 100
POP
GETLOCAL i               // i インクリメント
PUSH 1
ADD
SETLOCAL i               // for 脱出判定
GETLOCAL ForEndNum
GETLOCAL i
SUB
LOGNOT
JPEQ0 BeginForLabel
END
```

結果

for 文反復回数10^6 以降は VM の実行時間が測定不能なほど大きくなった。

C++に比べてVMの方が増加し始めるのが早い。



C++実装							
	100	1000	10000	100000	1000000	10000000	100000000
1	11.5566	10.6603	11.5778	9.7309	12.7215	25.2537	154.587
2	10.4608	11.0242	10.3316	9.9442	12.3759	24.2626	153.1485
3	11.0574	10.2941	12.9122	11.2993	11.7711	24.6246	154.8835
4	9.97	10.323	13.2096	12.3754	11.9906	27.2948	154.6893
5	10.7022	10.6007	11.137	11.9263	11.8278	25.4812	153.7658
6	10.7727	10.2179	10.7184	11.1748	11.5187	24.2349	159.0125
7	10.8675	11.0811	10.9372	11.387	12.7817	25.0401	154.1233
8	10.4428	10.3831	10.6713	11.8532	11.9903	25.9658	156.1517
9	11.4149	10.3876	10.6765	10.2361	11.4811	25.1293	155.1098
10	13.0524	10.4868	10.2187	10.565	12.3103	25.6424	152.7317
ave	11.02973	10.54588	11.23903	11.04922	12.0769	25.29294	154.82031

VM実装							
	100	1000	10000	100000	1000000	10000000	100000000
1	12.273	26.0933	184.0079	1735.121			
2	11.7997	26.3467	183.6685	1752.868			
3	12.7249	25.5928	183.6235	1740.341			
4	13.7449	25.8745	184.4553	1746.919			
5	13.8141	27.7909	184.7542	1749.279			
6	12.6699	25.5153	183.8948	1765.235			
7	12.7557	30.227	184.5262	1753.07			
8	13.9078	26.7192	185.1946	1742.373			
9	14.8731	26.0598	184.1016	1747.189			
10	12.7219	25.4469	183.4803	1746.186			
ave	13.1285	26.56664	184.1707	1747.858	N/A	N/A	N/A

考察

・VM実装について、現状線形的に実行時間が増加しているため、反復回数が増加するにつれ今後も同様の増加傾向になると考えられる。

・命令実行のたびに命令解釈を行うため、VM のほうが1ループに対する実行時間コストが大きい。そのため反復回数に対する時間増加がより少ない反復回数から起きはじめたと考えられる。

今後の展望

・命令の解釈をした後の情報を保存しておくことによる、命令解釈の実行時間の短縮。

・初期化の時間などを省いたループのみの時間測定
　→ VM での時間測定機構が必要

質問