Project 3

컴퓨터의 개념 및 실습

Computer Concepts & Practice

2023. 5. 23

박준영

Virtual Machine & Optimization Laboratory

Dept. of Electrical and Computer Engineering

Seoul National University

Computer

규칙이 정해진 계산기

- 외부에서 데이터를 받아, 정해진 규칙(프로그램)대로 계산하고, 외부에 데이터를 다시 돌려주는 장치

프로그램 == 컴퓨터

- 다른 컴퓨터(프로그램)를 흉내내는(실행하는) 컴퓨터
  - 범용 튜링 머신(universal Turing machine)

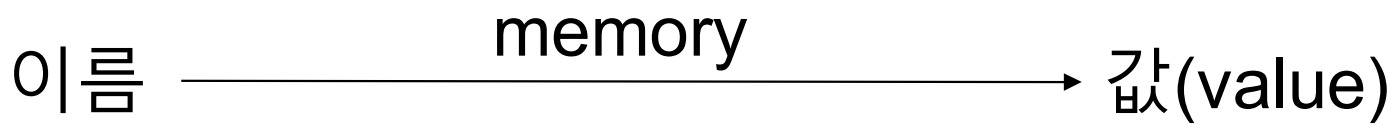
Computer

구성요소

- 실행할 명령(프로그램)
- 내부 상태(state(context), memory)

메모리(memory)

- 변수의 이름을 받아 그 값을 돌려주는 함수
  - x → 4, my_list → [1, 2, 3], z → (240, 12, 32)



Virtual Machine

주로 소프트웨어로 구현한 가상의 컴퓨터

- 컴퓨터 안의 컴퓨터(프로그램)

H/W와 독립적인 환경을 제공하기 위해 사용하기도 함

- Java의 JVM(Hotspot, Dalvik), .NET, Python VM (PVM)

컴퓨터 시스템 전부를 흉내내는 경우

- 시스템 VM: VMWare, Virtual Box, KVM, QEMU, …

Petite Language / Plang

PROJECT 3

Programming Language

컴퓨터와 소통하는 방식

- 컴퓨터에게 해야 할 일의 순서를 알려주거나
- 해야 할 일을 알려주거나

명령의 생김새와 그 행동으로 구성

- 생김새(syntax)
  - ‘x = 5’ → 맨 처음에는 변수의 이름(id), 그리고 ‘=‘, 마지막으로 숫자
- 해야 할 행동(semantics)
  - ‘x = 5’ → 변수 ‘x’를 없다면 만들어서, ‘x’가 ‘5’의 값을 지니게 한다

Petite Language

Example: 피보나치 수열 계산하기

```
    jmp 1, MAIN
    # 1 1 2 3 5 8 13 21 34 55 …

MAIN:
    x = input()
    memo = [1; x]
    i = 2
LOOP:
    memo[i] = memo[i - 1] + memo[i - 2]
    i = i + 1
    jmp i < x, LOOP

    i = 0
PRINT_LOOP:
    print(memo[i])
    i = i + 1
    jmp i < x, PRINT_LOOP
```

Petite Language

작은 언어

- 실수형(float) 없이, 오직 정수형(int)만 사용
- 가능한 명령은 오직 6종류

작지만 유능한

- 다른 프로그래밍 언어들이 할 수 있는 일을 모두 할 수 있음
  - 튜링 완전

- 언어의 형태와 표현력
  - 모든 일을 할 수 있지만, 하기 쉬운 건 아님
    e.g., Plang으로 그림 그리기

  - “말하는 대로 생각하게 된다”

Petite Language

명령 (Command)

- 한 줄에 하나의 명령이 있을 수 있음
  - 빈 줄도 가능

- **LABEL:**
  - 현재 위치를 'LABEL'이라는 이름으로 정의
    'HERE:', 'LOOP:', 'PREHEADER:', 'SCREAMING_SNAKE_CASE'
  - 이 명령줄(command-line)의 '라벨'과 '줄 번호'를 메모리에 저장
  - 'INT_TYPE'이 아닌 **'int'형**으로 저장할 것!

Petite Language

명령 (Command)

- `jmp E, LABEL`
  - 'E'의 값이 0이 아니라면, 다음에는 'LABEL'로 가서 실행
  - 'E'의 값이 0이라면, 바로 다음 줄 실행

  - 'E'는 list가 아니라 숫자여야 함
    숫자가 아니라면 'Illegal Value' 예외

  - 'LABEL'은 이미 정의된 라벨이어야 함
    찾을 수 없다면 'Unknown Label' 예외

Petite Language

명령 (Command)

- `print(E)`
  - 'E'를 출력
  - 'E'는 정수, list 모두 가능

```
    # Factorial 5
    x = 5
    z = 1
LOOP:
    z = z * x
    x = x - 1
    jmp x != 0, LOOP

    print(z)
```

Petite Language

명령 (Command)

- `x = E`
  - 'E'는 정수
    아닌 경우 '`IllegalValue`' 예외

- `x = [E1; E2]`
  - 'E1'이 'E2' 개 연속해 있는 list를 x에 저장
  - 'E1', 'E2'는 모두 정수
    아닌 경우 '`IllegalValue`' 예외

- `x[E1] = E2`
  - 'x'는 list
    아닌 경우 '`IllegalValue`' 예외
  - 'E1', 'E2'는 모두 정수
    아닌 경우 '`IllegalValue`' 예외
  - 'x'는 이미 정의된 list 변수
    메모리에 없는 경우 '`UnknownVariable`' 예외

Petite Language

Example: 피보나치 수열 계산하기

항상 jump!

```
jmp 1, MAIN
# 1 1 2 3 5 8 13 21 34 55 …

MAIN:
    x = 10
    memo = [1; x]
    i = 2
```

```
for i in range(2, x):
    memo[i] = memo[i - 1] + memo[i - 2]
```

```
LOOP:
    memo[i] = memo[i - 1] + memo[i - 2]
    i = i + 1
    jmp i < x, LOOP
```

```
for i in range(x):
    print(memo[i])
```

```
    i = 0
PRINT_LOOP:
    print(memo[i])
    i = i + 1
    jmp i < x, PRINT_LOOP
```

Petite Language

표현식 (Expression)

- ‘E’에 들어갈 수 있는 식
- **n**
  - 임의의 10진수 숫자 (음수 가능)

    0, 4, 121687, 19937, -81, -7

- **x**
  - 소문자 알파벳 혹은 ‘_’(underscore)로 구성된 이름이어야 함

    `x, y, my_age, learning_rate, snake_case`
  - 이미 예약된 단어들(keyword)과 같을 수 없음

    `‘jmp’, ‘print’, ‘input’`

- **x[E]**
  - list 변수 ‘x’의 index-‘E’의 원소

Petite Language

표현식 (Expression)

- **(E)**
  - 우선순위를 갖는 E
  - 'E'랑 동일해야 함

- **`input()`**
  - 키보드로 정수를 입력 받음
    앞에 'INPUT: '이 출력되야 함

- **E1 op E2**
  - 이항 연산자로 묶인 두 표현식
    `x + 5, 4 * 8, 9 - 2, 7 / 3, 1 == 0, x <= 42`

Petite Language

연산자 (Operator)

- ‘E1 **op** E2’의 **op**
- 두 개의 입력(항)을 받는 연산자를 이항 연산자라 함

사칙연산

- **+**
- **-**
  - 항이 항상 두 개 필요. 단항(unary) ‘-’는 없음 (`-height` (X), `0 - height` (O))
- ***
- **/**
  - 정수 나눗셈 몫을 반환해야 함
  - 파이썬의 정수 나눗셈(floor)과 달리 소수점을 모두 버림(truncate)

    -999 / 10 → **-99** (C/C++, Plang) -999 // 10 → -100 (Python)

Petite Language

연산자 (Operator)

- ‘E1 **op** E2’의 **op**
- 두 개의 입력(항)을 받는 연산자를 이항 연산자라 함

관계연산 (Relational)

- **==**
- **!=**
- **<**
- **<=**
- **>**
- **>=**
  - 관계 연산자들은 모두 참일 때 1, 거짓일 때 0
  - **TIP!** 1과 0의 타입에 대해 주의할 것

Petite Language

들여쓰기 (Indentation)

- 라벨 정의를 제외한 명령들은 모두 4칸 들여 씀
- 언어의 문법 규정이 아닌 가독성을 위한 약속 (Convention)

띄어쓰기 (Whitespace)

- 소괄호, 대괄호를 제외한 연산자는 피연산자와 모두 띄어 씀
  - e.g., 4 + 1, x + (y * 3), x[15]
- 'input()'의 경우 모두 붙여서 써야 함
- 'print(…)'의 경우 시작하는 소괄호가 'print'와 붙어 있어야 함
- 쉼표(,), 세미콜론(;)은 앞에 붙여 씀
  - e.g., jmp 1, MAIN x = [0; 12]

주석 (Comment)

- '#' 부터 해당 줄의 끝까지는 주석

Petite-VM

PROJECT 3

Goals

표현식 변환기 (30 pt.)

- 실습시간에 배운 '차량기지 알고리듬'을 사용해 중위 표현식을 후위 표현식으로 고쳐주는 함수 `in2post`를 구현하세요.

Petite-VM (60 pt.)

- Plang (petite-language)을 읽고 동작하는 VM, `PTVM`을 구현하세요.

(Bonus) 고정 길이 정수 (20 pt.)

- 비트 길이가 정해진 정수를 위한 `class ixx`를 구현하세요.
  - 최대 20점의 보너스 점수가 주어지지만 점수의 합은 100을 넘을 수 없습니다.

in2post

표현식 변환

- 중위 표현식을 후위 표현식으로 바꿔주는 함수 in2post를 구현하세요.

연산자 테이블

- 연산자에 해당하는 기호와 함수가 정의된 테이블이 매개변수로 제공됨
- 소괄호(‘( )’)와 대괄호(‘[ ]’)는 특수하게 처리 해야함
- 소괄호가 제대로 열리고 닫히지 않은 경우 ‘`MismatchingParentheses`’ 예외 발생
- 대괄호가 제대로 열리고 닫히지 않은 경우 ‘`MismatchingBrackets`’ 예외 발생
  - 둘 모두 가능하다 판단하면 소괄호 우선

- 여는 대괄호(‘[‘) 기호를 __getitem__에 해당하는 연산 기호로 취급함
  - e.g., x[4] → x 4 [

VM

기본적인 VM의 형태를 제공

- pc: 이번에 실행할 코드의 줄 번호 (0부터 시작함)
- mem: VM의 내부 메모리, 상태
- code: VM이 실행할 코드 조각
- code_lines: 코드조각에 담긴 명령 줄의 수, 길이
- operators: 연산자 테이블

evaluate

- 표현식, E의 값을 계산하는 메소드

calc

- 후위 표현식, ‘postfix_expr’의 값을 계산하는 메소드

compute

- 이항연산자들로 묶인 표현식의 값을 계산하는 메소드

PTVM

Petite-VM, PTVM

- Plang으로 작성된 코드를 읽어 실행하는 VM, class PTVM을 구현 하세요.

INT_TYPE

- ‘ixx.py’에서 구현한 고정 길이 정수형을 사용할 수 있어야 함
  - i.e., i8, i16, i32, i64

연산자 테이블

- Plang의 연산자들이 우선순위 순서대로 {‘기호’: 함수}의 list로 주어짐

부모 class

- 가상머신을 위한 class VM을 상속받음

PTVM

예외 (Exception)

- Handout 참고
- 모든 예외는 예외가 발생하는 명령줄이 끝나기 전에 발생해야 함
  - Plang 프로그램의 실행이 모두 끝나고 예외 발생 → 잘못된 구현

(Bonus) class ixx

고정 길이 정수

- 비트 길이가 정해진 정수를 위한 `class ixx`를 구현하세요.

제약 조건

- 정수를 2진수로 바꿔주는 int(…, 2) 혹은 format(…, b)은 사용할 수 없음

- 연산자를 구현할 때에는 'int'의 해당 연산자 값을 반환할 수 없음
  - e.g., '__add__'를 구현할 때, '`ixx(int(self) + int(other))`'를 반환
  - 중간 계산 과정에서 필요한 경우 사용 가능
  - 구현하는 '`ixx`'의 연산자는 사용 가능

(Bonus) class ixx

구현목표

- __init__
- __int__
- __neg__
- __add__
- __sub__
- __mul__
- __truediv__
- __eq__
- __ne__
- __lt__
- __gt__
- __le__
- __ge__

(Bonus) class ixx

__init__

- 10진 정수 입력이 가진 비트 수로 표현이 불가능한 경우
  - ‘OutOfCoverage‘ 예외 발생
- 비트열(벡터) 입력에 0 혹은 1이 아닌 수가 있는 경우
  - ‘IllegalBitVector’ 예외 발생
- 비트열(벡터) 입력이 가진 비트 수 보다 긴 경우
  - ‘OutOfCoverage‘ 예외 발생
- 입력을 정수형(int)로 바꿀 수 없는 경우
  - ‘IllegalType’ 예외 발생

(Bonus) class ixx

__int__

- 가진 비트열의 값을 10진 정수로 변환해서 반환

__neg__

- 부호를 음수로 전환

__add__, __sub__, __mul__

- 'other'의 타입이 'ixx' 계열이 아닌 경우 'IllegalType' 예외 발생
- 계산 결과가 비트 범위를 벗어나면 'Overflow' 예외 발생

__truediv__

- 정수 나눗셈의 몫을 반환
- 'other'의 타입이 'ixx' 계열이 아닌 경우 'IllegalType' 예외 발생
- 0으로 나누는 경우 'DivideByZero' 예외 발생

(Bonus) class ixx

__bool__

- 가진 값이 0에 해당하면 False, 그 외의 모든 숫자는 True

__abs__

- 현재 숫자의 절댓값을 'ixx'형태로 반환

(Bonus) class ixx

__eq__, __ne__, __lt__, __gt__, __le__, __ge__

- 조건이 참이면 'self'와 동일한 타입으로 1 반환
- 조건이 거짓이면 'self'와 동일한 타입으로 0 반환

주어지는 함수

- larger_type
  - 두 'ixx'변수의 타입 중 비트 길이가 더 긴 타입을 반환
- zext
  - 부족한 자리들을 0으로 채워주는 함수
- sext
  - 부족한 자리들을 가장 큰 비트로 채워주는 함수 (부호 유지)

PTVM

참고사항

- '**virtual_machine.py**', '**__init__.py**', 각 class의 '**__repr__**' 및 이미 존재하는 **instance variable**들을 **제외**한 모든 파일, 모든 위치를 **수정 가능**
- 설치 없이 사용 가능한 모든 기본 파이썬 모듈 import 가능
  - import math (O), import numpy (X)
- 'example' 디렉토리에 Plang 예제 제공
- 채점 시 출력(print(…))이 아닌, PTVM의 상태(memory, pc값)를 비교할 예정
  - 주어진 코드를 정확한 시점에 정확하게 실행해야 함
- 문법은 띄어쓰기를 깐깐하게 봄
  - 'x = 4' (O), 'x=4' (X)
- Handout을 꼭 읽어볼 것
  - 이 슬라이드가 모든 내용을 설명하고 있지 않음
  - 반대로 handout도 모든 내용을 설명하고 있지 않음

Submission

eTL 과제 'Project 3' 에 제출

- 'vm' 디렉토리를 zip 파일로 압축해서 'vm_{학번}_{이름}.zip'으로 제출
  - **vm_2023-12345_홍길동.zip**

제출 기한

- 2023년 **6월 9일** 자정 전(23:59) 까지 제출
- **Delay**: 하루당 -10% (최대 3일, -30%) 감점, (6/12 월요일 까지)

Cheating, Copy 금지

- 적발 시 강력한 penalty 부여 예정
- 어렵고 생소한 과제이므로 동료들 간의 의견교환 적극 권장
  - '이렇게 해보면 잘 될 거야' **(O)**, '나는 이렇게 짰어 (코드를 보여준다)' **(X)**
    말로도 코드를 직접 알려주지 말 것!

eTL 질의 응답 게시판 사용, 확인 권장