Grundbegriffe der Informatik

Übung

Simon Wacker

Karlsruher Institut für Technologie

Wintersemester 2015/2016

Programmiersprachen

Formale Sprachen deren Wörter Programme sind

Programm in Maschinensprache

- Folge von Bits
- direkt von Maschine ausführbar

Programm in Assemblersprache

Programm in höherer Programmiersprache

Programmiersprachen

Formale Sprachen deren Wörter Programme sind

Programm in Maschinensprache

Programm in Assemblersprache

- Folge von Zeichen
- hardwarenah
- nach Assemblierung von Maschine ausführbar
- `STV 42` wird zu `00100000000000000101010`

Programm in höherer Programmiersprache

Programmiersprachen

Formale Sprachen deren Wörter Programme sind

Programm in Maschinensprache

Programm in Assemblersprache

Programm in höherer Programmiersprache

- Folge von Zeichen
- hoher Abstraktionsgrad
- nach Compilierung von Maschine ausführbar
  - oft in zwei Schritten
  - erst zu Programm in Assemblersprache
  - dann zu Programm in Maschinensprache
  - `x = 42` wird zu `LDC 42; STV A(x)` wird zu `...`

Kontrollstrukturen

Steuern Ablauf eines Programms in imperativen Programmiersprachen

In Assemblersprachen

- Unbedingter Sprung: `JMP` a
- Bedingter Sprung: `JMN` a

In höheren Programmiersprachen

- Bedingte Anweisung: `if` B `do` S `end`
- Verzweigung: `if` B `do` S `else` T `end`
- Kopfgesteuerte Schleife: `while` B `do` S `end`
- Fußgesteuerte Schleife: `repeat` S `until` B `end`
- Zählschleife: `for` i `from` x `to` y `do` S `end`

Übersetzung in Assemblersprache?

Bedingte Anweisung

```
if M(a1) = M(a2) then
   S
end
U
```

```
if: LDV  a1
    EQL  a2
    NOT
    JMN  end
then: S
end: U
```

Verzweigung

```
if M(a1) = M(a2) then
   S
else
   T
end
U
```

```
if:   LDV  a1
      EQL  a2
      JMN  then
else: T
      JMP  end
then: S
end:  U
```

Kopfgesteuerte Schleife

```
while M(a1) = M(a2) do
   S
end
T
```

```
while: LDV  a1
       EQL  a2
       NOT
       JMN  end
  do:  S
       JMP  while
 end:  T
```

Kopfgesteuerte Schleife — continue

```
while M(a1) = M(a2) do
   S1
   if M(a3) = M(a4) then
      continue
   end
   S2
end
T
```

```
while: LDV  a1
       EQL  a2
       NOT
       JMN  end
  do:  S1
  if:  LDV  a3
       EQL  a4
       NOT
       JMN  next
then:  JMP  while
next:  S2
       JMP  while
 end:  T
```

Kopfgesteuerte Schleife — break

```
while M(a1) = M(a2) do
  S1
  if M(a3) = M(a4) then
    break
  end
  S2
end
T
```

```
while: LDV  a1
       EQL  a2
       NOT
       JMN  end
  do:  S1
  if:  LDV  a3
       EQL  a4
       NOT
       JMN  next
then:  JMP  end
next:  S2
       JMP  while
 end:  T
```

Fußgesteuerte Schleife

```
repeat
   S
until M(a1) = M(a2) end
T
```

```
repeat: S
 until: LDV  a1
        EQL  a2
        NOT
        JMN  repeat
   end: T
```

Datenstrukturen

Speichern und organisieren Daten

In höheren Programmiersprachen

- Vektoren oder eindimensionale Arrays
- Matrizen oder mehrdimensionale Arrays
- Verkettete Listen
- Graphen
- Hashtabellen

Vektoren in Minimalmaschine: Siehe Vorlesung

Matrizen in Minimalmaschine?

Matrix

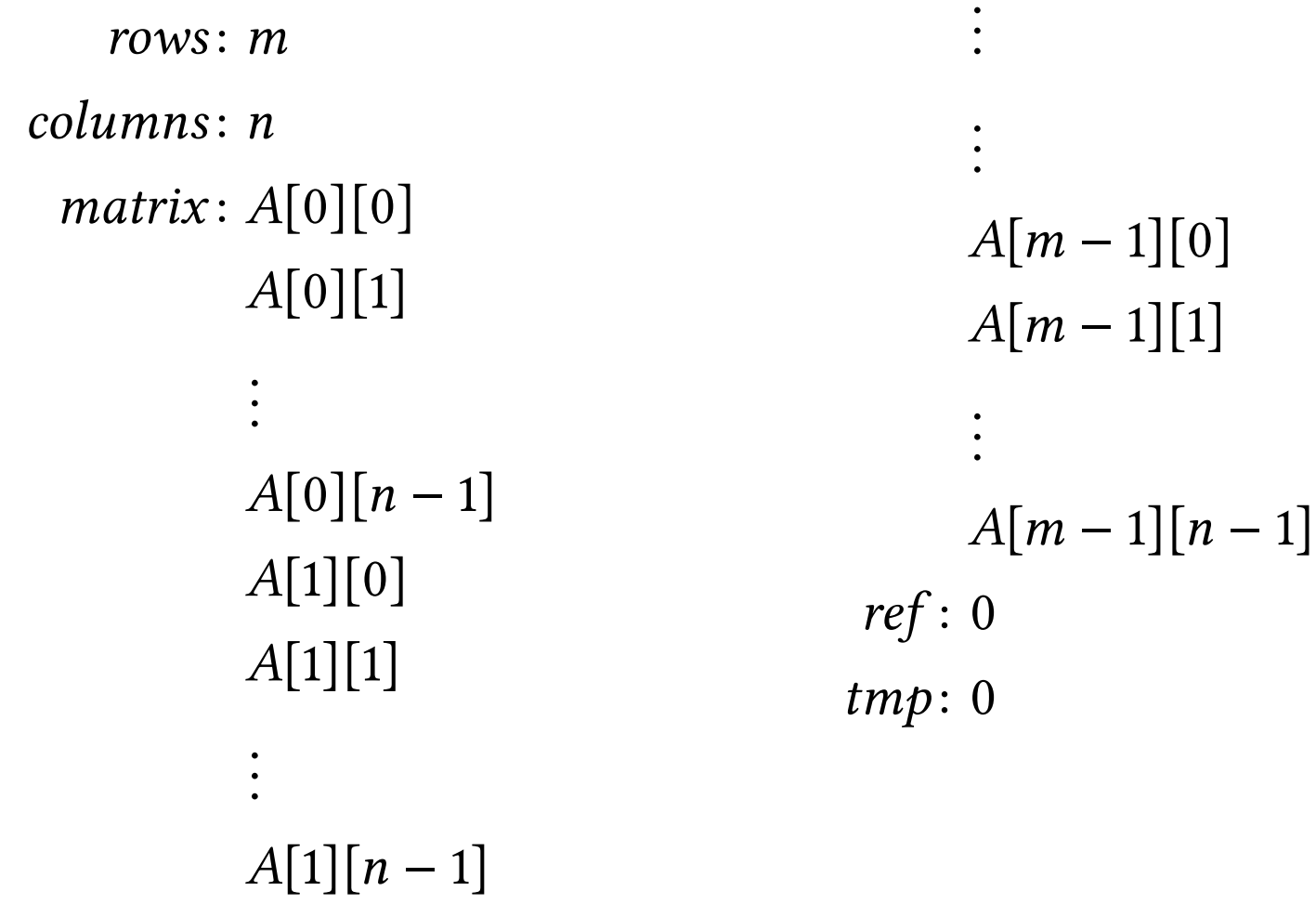
A Matrix mit $m \in \mathbb{N}_+$ Zeilen und $n \in \mathbb{N}_+$ Spalten

Einträge von A ganze Zahlen, die mit 24bit in Zweierkomplementdarstellung darstellbar

$A[i][j]$ Eintrag in i-ter Zeile und j-ter Spalte, wobei $i \in \mathbb{Z}_m$ und $j \in \mathbb{Z}_n$

Im Speicher legen wir die Einträge von A zeilenweise ab

Matrix — Speicher



Matrix — Eintrag laden

Adresse in der $A[i][j]$ steht, $matrix + i \cdot n + j$, berechnen
$A[i][j]$ mit indirekter Adressierung laden

```
i_times_n: LDC  i
           STV  tmp
while:     LDC  0
           NOT
           ADD  tmp
           STV  tmp
           JMN  plus_j
do:        LDV  columns
           ADD  ref
           STV  ref
           JMP  while
plus_j:    LDC  j
           ADD  ref
           STV  ref
plus_matrix: LDC  matrix
           ADD  ref
           STV  ref
load:      LDIV ref
```