# Einführung in die Rechnerarchitektur (ERA) IN0004

## Sprünge und Unterprogramme

**Martin Schulz**
**schulzm@in.tum.de**
Chair for Computer Architecture and Parallel Systems
https://www.ce.cit.tum.de/caps/

**Robert Wille**
**robert.wille@tum.de**
Chair for Design Automation
https://www.cda.cit.tum.de/

# Steuerung des Programmablaufs: Unterprogrammaufruf

Bisher

- Sequentielle Programmabfolge
- Sprünge sind möglich → Schleifen
- Monolithischer Block

Was fehlt noch?

- Aufruf von gemeinsamer Funktionalität
  - Strukturierte Programmierung
  - Bibliotheken/Libraries von anderen ProgrammiererInnen

Unterprogrammaufruf

- Sprung an eine Routine
- Vor dem Sprung: Sichern der Rückkehradresse
- Am Ende der Routine, Rücksprung an diese Adresse
- Konventionen um sicher zu stellen, dass nichts überschrieben wird
  - Welche Information muss gesichert werden, welche kann überschrieben werden?
  - Beliebige Routinen müssen zusammenpassen können

Wiederholung

# Unterprogramme in RISC-V

Begriffe

Caller = Aufrufende Funktion (im Beispiel `main`)
Callee = Aufgerufene Funktion (im Beispiel `simple`)

**`jal reg, offset:`** `reg` = PC + 4 (0x00000304) bzw. nächster Befehl
Sprung zu PC+offset (im Beispiel label **`simple`** bei 0x0000051c)

Rücksprung: **`jr ra`** PC = `ra` (0x00000304)

**C Code**

```
int main() {
  simple();
  a = b + c;
}

void simple() {
  return;
}
```

**RISC-V Assembler**

```
0x00000300 main:   jal  simple      # call
0x00000304         add  s0, s1, s2
...                ...

0x0000051c simple: jr   ra          # return
```

Wiederholung

# Anmerkungen

Sprünge und Unterprogramm Aufrufe sind die gleichen Befehle

```
j offset          =     jal x0,offset
jr reg, offset    =     jalr x0,reg,offset
```

→ **Pseudobefehle / “Pseudo Instructions”**
(werden vom Assembler automatisch ersetzt)

Weiterer Pseudo-Befehl

`mv ra,rb` = `add ra,rb,x0` oder `addi ra,rb,0`

Andere ISAs haben hier Spezialbefehle

- Call Befehle
- Return Befehle

- Implizite Speicherung der Rücksprungadresse
  (bei RISC-V ist das explizit)

Wiederholung

# Argumente und Rückgabewert (1)

## C Code

```
int main()
{
  int y;
  ...
  y = diffofsums(2, 3, 4, 5);  // 4 arguments
  ...
}

int diffofsums(int f, int g, int h, int i)
{
  int result;
  result = (f + g) - (h + i);
  return result;                // return value
}
```

Wiederholung

# Aufrufkonvention

Soviel wie möglich via Register

- Eingabedaten
- Rückgabewert

Gezielte Aufgaben

- a0-a7: Eingaben
- a0-a1: Ergebnisse
- ra: Rücksprungadresse

Caller-saved

- Aufrufer/Caller muss Werte selbst sichern, dürfen von Callee verändert werden

Callee-saved

- Aufgerufene Funktion darf Werte nicht verändern oder muss sie wieder herstellen

| Register | ABI Name | Description | Saver |
|---|---|---|---|
| x0 | zero | Hard-wired zero | — |
| x1 | ra | Return address | Caller |
| x2 | sp | Stack pointer | Callee |
| x3 | gp | Global pointer | — |
| x4 | tp | Thread pointer | — |
| x5–7 | t0–2 | Temporaries | Caller |
| x8 | s0/fp | Saved register/frame pointer | Callee |
| x9 | s1 | Saved register | Callee |
| x10–11 | a0–1 | Function arguments/return values | Caller |
| x12–17 | a2–7 | Function arguments | Caller |
| x18–27 | s2–11 | Saved registers | Callee |
| x28–31 | t3–6 | Temporaries | Caller |
| f0–7 | ft0–7 | FP temporaries | Caller |
| f8–9 | fs0–1 | FP saved registers | Callee |
| f10–11 | fa0–1 | FP arguments/return values | Caller |
| f12–17 | fa2–7 | FP arguments | Caller |
| f18–27 | fs2–11 | FP saved registers | Call |
| f28–31 | ft8–11 | FP temporaries | |

From: https://riscv.org/technical/specifications/

Wiederholung

# Argumente und Rückgabewert (2)

## RISC-V Assembler

```
main:
addi a0, zero, 2   # Argument 0 = 2
addi a1, zero, 3   # Argument 1 = 3
addi a2, zero, 4   # Argument 2 = 4
addi a3, zero, 5   # Argument 3 = 5
jal  diffofsums    # Aufruf
add  s7, a0, zero # Variable y in s7
```

```
diffofsums:
add  t0, a0, a1    # t0 = f + g
add  t1, a2, a3    # t1 = h + i
sub  t2, t0, t1    # result = (f + g) - (h + i)
add  a0, t2, zero # Ergebnis in a0 speichern
jr   ra            # Rücksprung
```

```
...
y = diffofsums(2, 3, 4, 5);
...

int diffofsums(int f, int g, int h, int i)
{
  int result;
  result = (f + g) - (h + i);
  return result;
}
```

Register t0,t1,t2
werden überschrieben!

Wiederholung

# Bedarf an weiteren Speichermöglichkeiten

Anzahl der Register ist immer limitiert

- Begrenzte Anzahl von Parametern
- Begrenzte Anzahl von temporären Werten
- Keine Möglichkeit Werte in temporären Registern zu sichern
- Aufruf von Unterprogrammen in Unterprogrammen
- Rekursion

Nötig Hauptspeicher zu verwenden

- Feste Speicherbereiche passen hier aber nicht
- Dynamischer Aufruf von ein oder mehreren Unterprogrammen

Temporäre Speicherbereiche

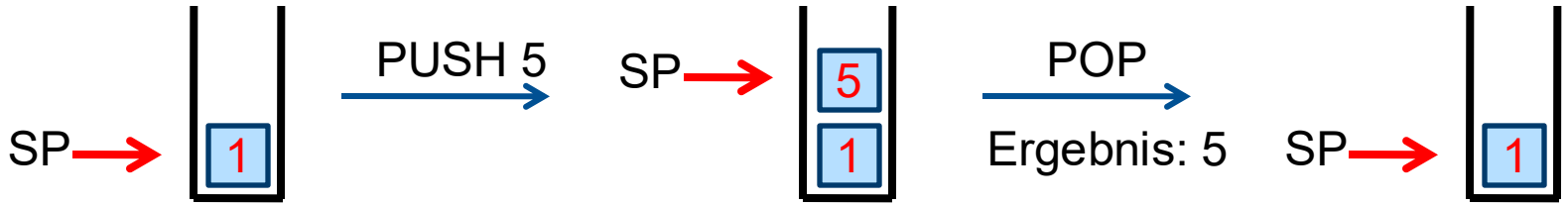
- Angelegt beim Unterprogrammaufruf
- Verfügbar um lokale Daten zu speichern
  - Weitere Parameter und temporäre Daten
  - Rücksprungadresse für diesen (einen) Aufruf
- Freigabe bei Rücksprung

# Passende Datenstruktur: Keller/Stapel/Stack

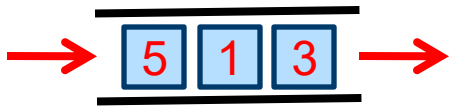
Liste von Zahlen, der man auf einer Seite

- Werte hinzufügen kann (PUSH)
- Werte entnehmen kann (POP)
- Meist übereinander gezeichnet
- Das Ende wird markiert durch eine Zeiger (Stack Pointer)

LIFO Prinzip: Last In First Out




Gegenteil:
FIFO Prinzip: First In First Out
Warteschlange/Queue

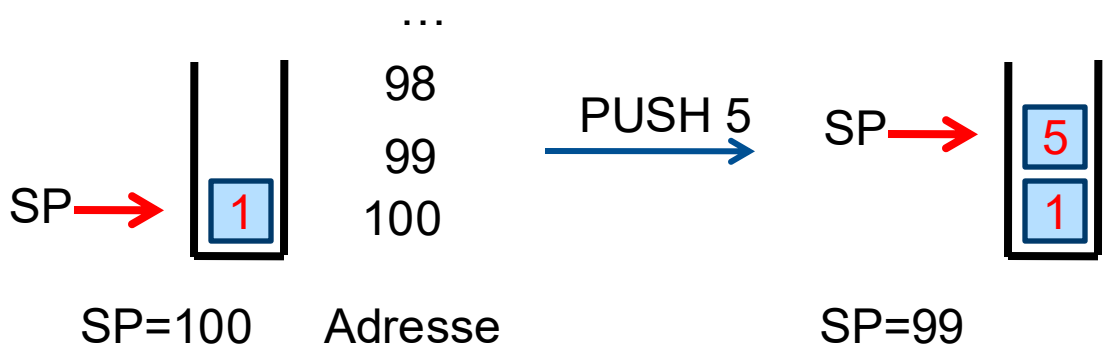


# Implementierung von Kellern/Stacks

Keller für Programmausführung werden oft im Speicher so abgelegt, dass sie „nach unten“ wachsen: Richtung kleinere Addressen

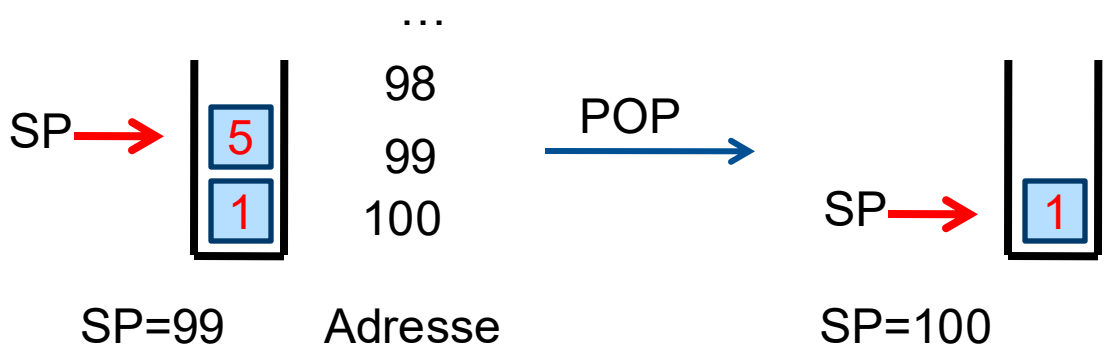
PUSH Funktionalität

- SP erniedrigen und in Adresse SP schreiben
- Prädekrement
- Viele ISAs haben dafür spezielle Befehle
- RISC-V
  - Anpassung von SP mit arithmetischen Befehlen
  - Schreiben mit „sd“ Anweisungen




POP Funktionalität

- Von Adresse SP lesen und SP erhöhen
- Postinkrement
- Viele ISAs haben dafür spezielle Befehle
- RISC-V
  - Lesen mit „ld“ Anweisungen
  - Anpassung von SP mit arithmetischen Befehlen

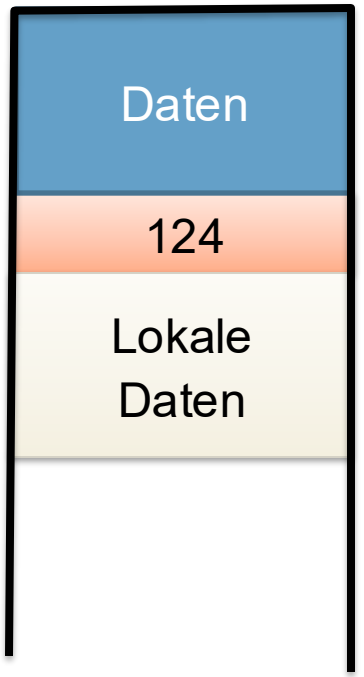


# Unterprogrammaufruf mit Kellern/Stacks

Unterprogramm

- Implementiert bestimmte Funktionalität
- Wird bei Bedarf aufgerufen
- Nach Ende, weiter Ausführung von der Stelle des Aufrufes

```
100: Work
…
123: Call compute (200)
124: Work
…

Compute:  200: get data
          201: do arithm
          …
          212: store data
          213: return
```




Genaue Details durch
Aufrufkonvention
(siehe oben)
definiert, als Teil der ISA

# Typische Speicherverwaltung

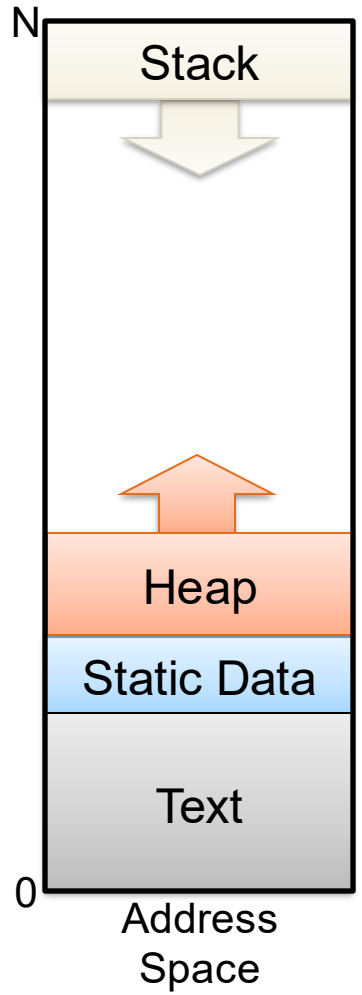
Jedes Programm hat einen Adressraum
- Enthält Daten und Programm

Code (”text”) wird meistens am unteren Ende geladen

Statische Daten
- Fest allozierte Daten als Teil des Codes

Dynamische Speicherverwaltung
- Auch “Heap” genannt
- Kann angefragt oder freigegeben werden
- Wächst nach oben

Keller/Stack wächst nach unten
- Optimale Platzausnutzung
- Wenn sie sich treffen -> Fehler



# Einführung in die Rechnerarchitektur (ERA) IN0004

## RISC-V Beispiel

**Martin Schulz**
**schulzm@in.tum.de**
Chair for Computer Architecture and Parallel Systems
https://www.ce.cit.tum.de/caps/

**Robert Wille**
**robert.wille@tum.de**
Chair for Design Automation
https://www.cda.cit.tum.de/

# Programm-Abstraktion

Eingabe: Code

- Kodiert als Text
- Nicht direkt ausführbar

C Code

```
int main(int argc, char *argv[])
{
   int c=0;
   for (int i=0; i<10; i++)
     { c = c + i;}
   printf("Ergebnis ist %i\n",c);
}
```

- Übersetzen in ausführbare Datei (Linux, z.B. Ubuntu)

```
>> gcc -o count -O0 -g count.c
```

- Ausführung des Programmes

```
>> ./count
Ergebnis ist 45
>>
```

**Compiler Optionen**

-o = Name der Ausgabedatei
-O<n> = Optimierungsgrad (0,1,2,…)
-g = Extra Symbole

Für weitere, siehe: man gcc

# Was ist in einer ausführbaren Binärdatei

Auslesen mit `objdump`

- Alle „Headers“ (Dateiteile) mit „-h“
- Codiertes Programm mit „-d“ = Dissassemblierung

```
0000000000000668 <main>:
 668: 7179            add sp,sp,-48
 66a: f406            sd ra,40(sp)
 66c: f022            sd s0,32(sp)
 66e: 1800            add s0,sp,48
 670: 87aa            mv a5,a0
 672: fcb43823        sd a1,-48(s0)
 676: fcf42e23        sw a5,-36(s0)
 67a: fe042423        sw zero,-24(s0)
 67e: fe042623        sw zero,-20(s0)
 682: a831            j 69e <main+0x36>
 684: fe842783        lw a5,-24(s0)
 688: 873e            mv a4,a5
 68a: fec42783        lw a5,-20(s0)
 68e: 9fb9            addw a5,a5,a4
 …
```

# Programm-Abstraktion

Eingabe: Code

- Kodiert als Text
- Nicht direkt ausführbar

C Code

```
int main(int argc, char *argv[])
{
    int c=0;
    for (int i=0; i<10; i++)
      { c = c + i;}
    printf("Ergebnis ist %i\n",c);
}
```

**Compiler Optionen**

-o = Name der Ausgabedatei
-O<n> = Optimierungsgrad (0,1,2,…)
-g = Extra Symbole

-march = ISA supported

- Übersetzen in ausführbare Datei (Linux, z.B. Ubuntu)

```
>> gcc -o count -O0 -g count.c -march=rv64imafd (statt rv64imafdc)
```

- Ausführung des Programmes

```
>> ./count
Ergebnis ist 45
>>
```

# Was ist in einer ausführbaren Binärdatei

Auslesen mit `objdump`

- Alle „Headers“ (Dateiteile) mit „-h“
- Codiertes Programm mit „-d“ = Dissassemblierung

```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
 674: 03010413          add s0,sp,48
 678: 00050793          mv a5,a0
 67c: fcb43823          sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
 684: fe042423          sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
 68c: 0240006f          j 6b0 <main+0x48>
 690: fe842783          lw a5,-24(s0)
 694: 00078713          mv a4,a5
 698: fec42783          lw a5,-20(s0)
 69c: 00f707bb          addw a5,a4,a5
 …
```

```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
 674: 03010413          add s0,sp,48
 678: 00050793          mv a5,a0
 67c: fcb43823          sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
 684: fe042423          sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
 68c: 0240006f          j 6b0 <main+0x48>
 690: fe842783          lw a5,-24(s0)
 694: 00078713          mv a4,a5
 698: fec42783          lw a5,-20(s0)
 69c: 00f707bb          addw a5,a4,a5
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
      00517             auipc a0,0x0
      50513             add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4:                   li a5,0
 6d8:                   mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          
 6e4: 03010113          
 6e8: 00008067          ret
```

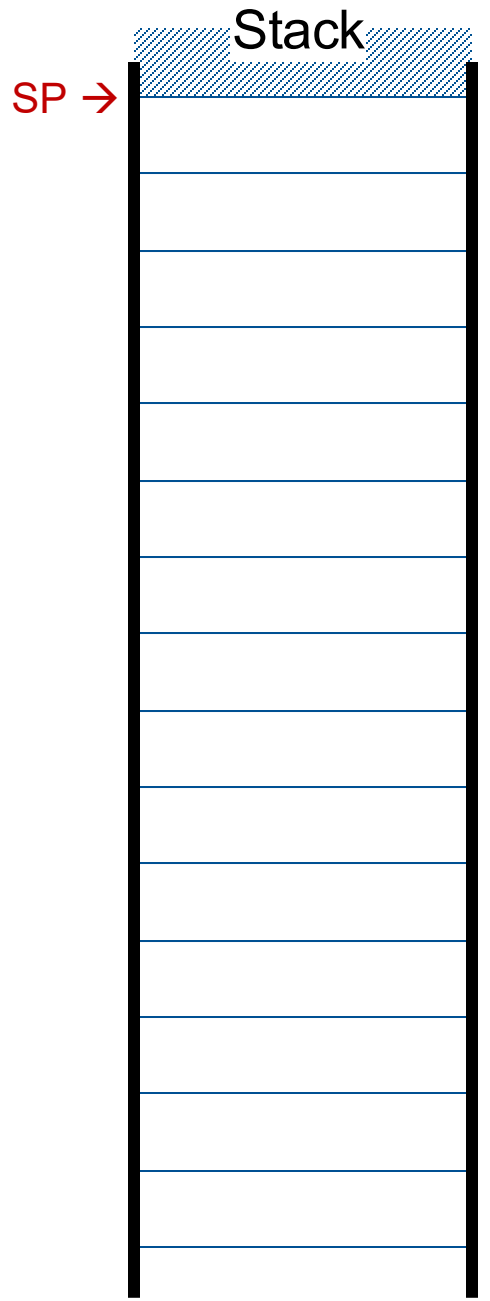
Adresse

Opcodes

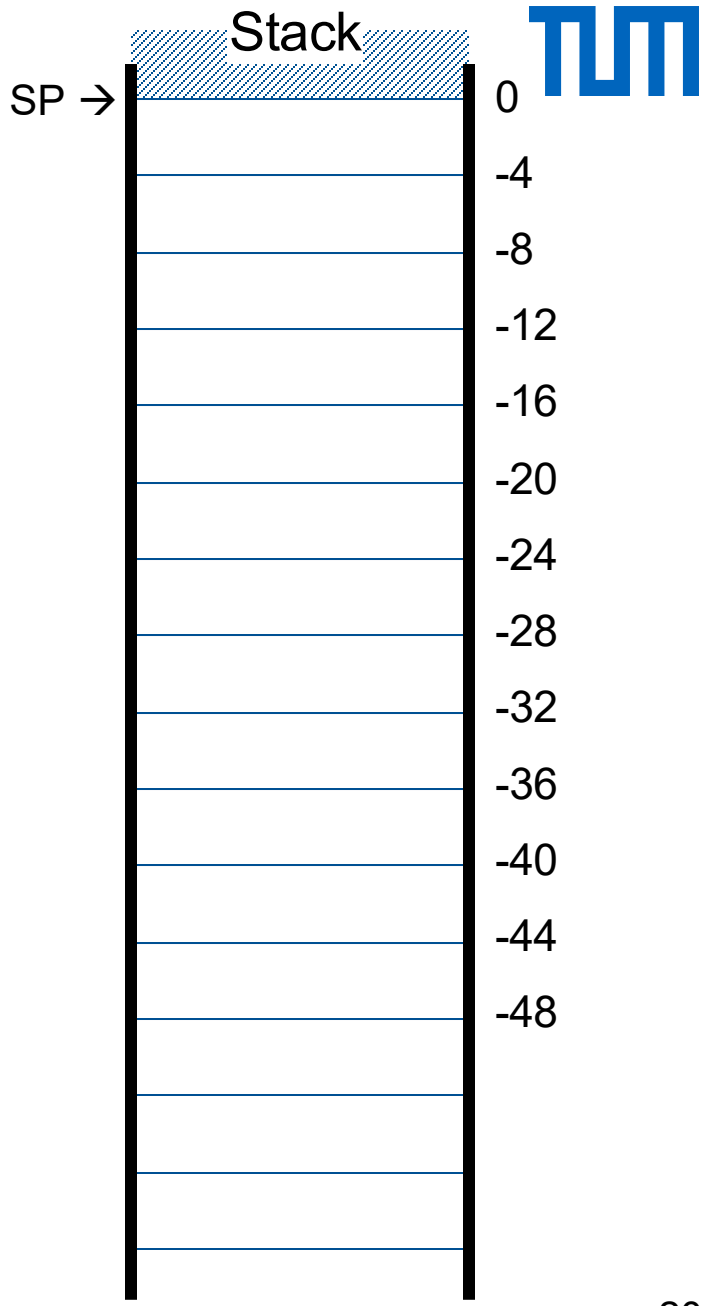
Assembler

```
0000000000000668 <main>:
PC→668: fd010113        add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
 674: 03010413          add s0,sp,48
 678: 00050793          mv a5,a0
 67c: fcb43823          sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
 684: fe042423          sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
 68c: 0240006f          j 6b0 <main+0x48>
 690: fe842783          lw a5,-24(s0)
 694: 00078713          mv a4,a5
 698: fec42783          lw a5,-20(s0)
 69c: 00f707bb          addw a5,a4,a5
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```

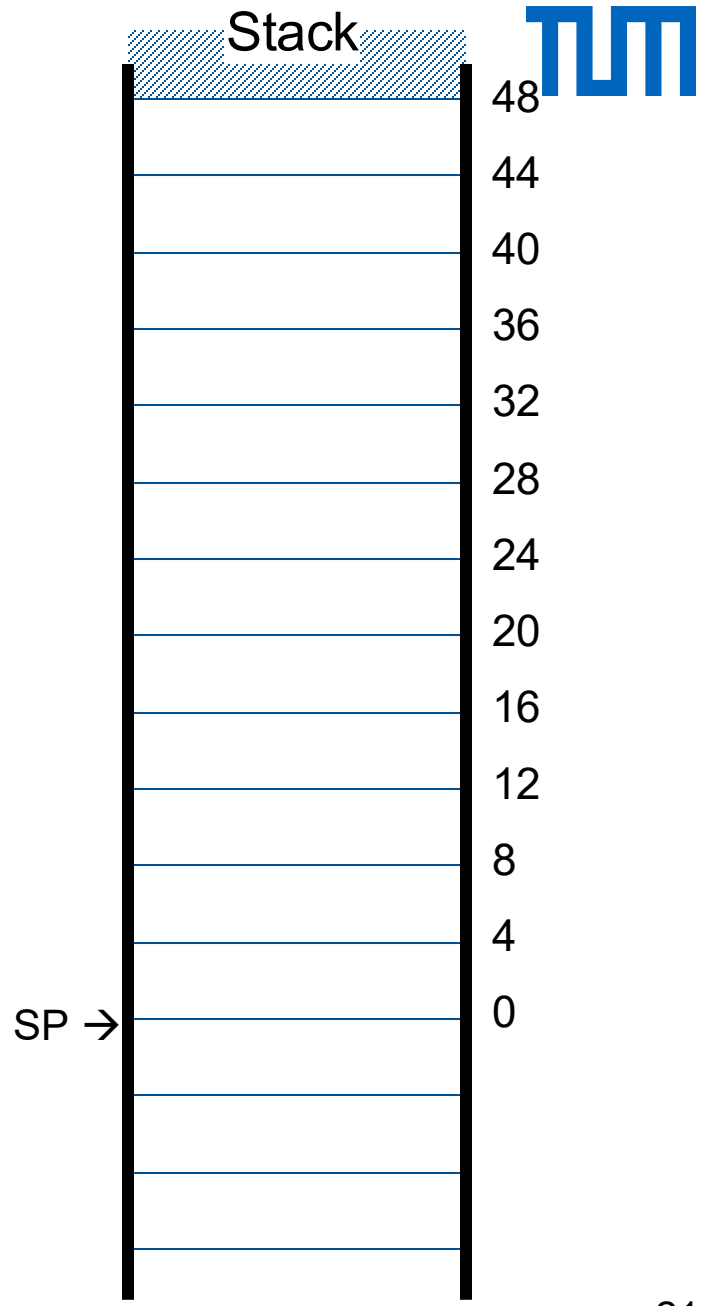
Function Prologue



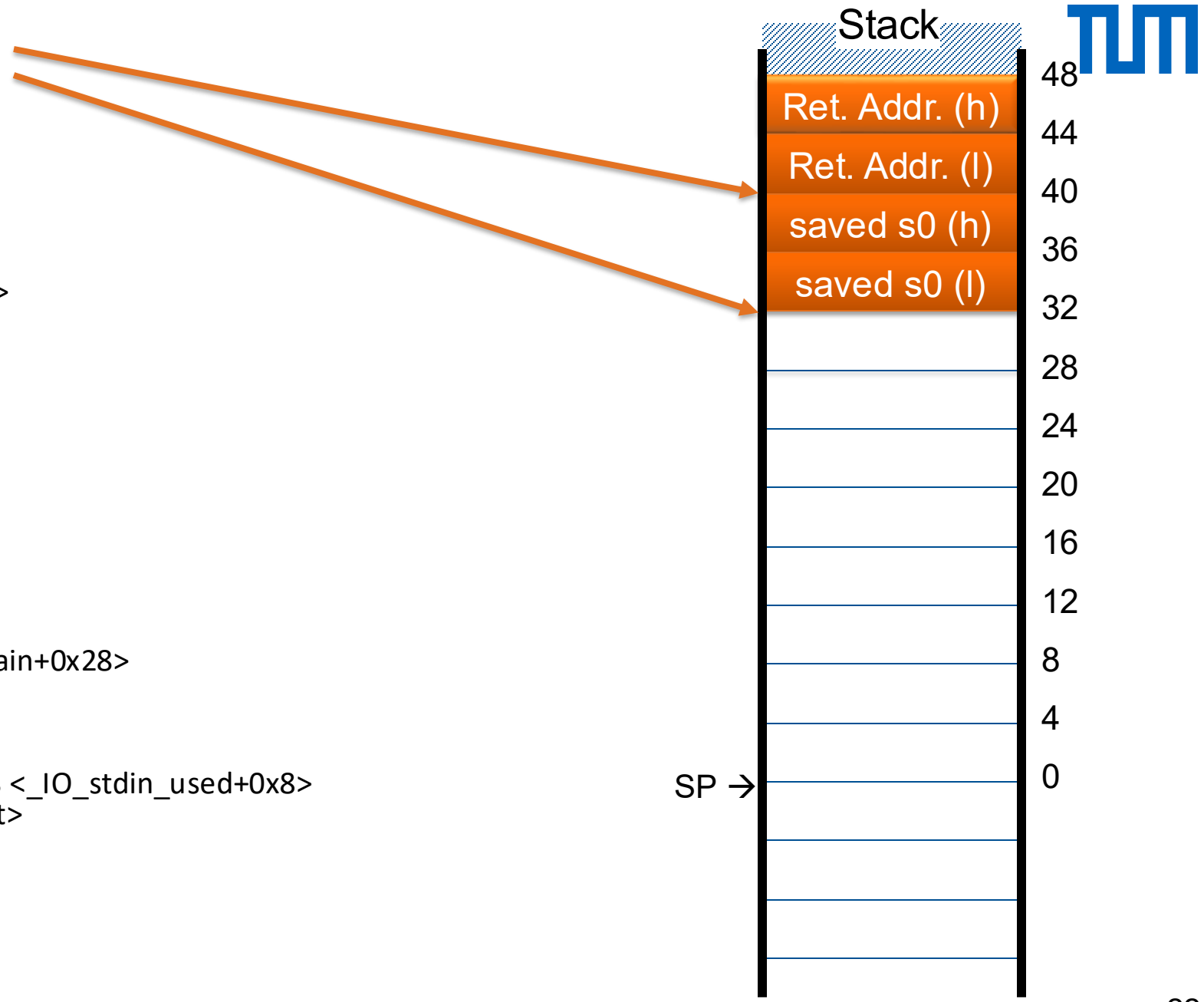
```
0000000000000668 <main>:
PC→668: fd010113          add sp,sp,-48
  66c: 02113423           sd ra,40(sp)
  670: 02813023           sd s0,32(sp)
  674: 03010413           add s0,sp,48
  678: 00050793           mv a5,a0
  67c: fcb43823           sd a1,-48(s0)
  680: fcf42e23           sw a5,-36(s0)
  684: fe042423           sw zero,-24(s0)
  688: fe042623           sw zero,-20(s0)
  68c: 0240006f           j 6b0 <main+0x48>
  690: fe842783           lw a5,-24(s0)
  694: 00078713           mv a4,a5
  698: fec42783           lw a5,-20(s0)
  69c: 00f707bb           addw a5,a4,a5
  6a0: fef42423           sw a5,-24(s0)
  6a4: fec42783           lw a5,-20(s0)
  6a8: 0017879b           addw a5,a5,1
  6ac: fef42623           sw a5,-20(s0)
  6b0: fec42783           lw a5,-20(s0)
  6b4: 0007871b           sext.w a4,a5
  6b8: 00900793           li a5,9
  6bc: fce7dae3           bge a5,a4,690 <main+0x28>
  6c0: fe842783           lw a5,-24(s0)
  6c4: 00078593           mv a1,a5
  6c8: 00000517           auipc a0,0x0
  6cc: 03050513           add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
  6d0: ed1ff0ef           jal 5a0 <printf@plt>
  6d4: 00000793           li a5,0
  6d8: 00078513           mv a0,a5
  6dc: 02813083           ld ra,40(sp)
  6e0: 02013403           ld s0,32(sp)
  6e4: 03010113           add sp,sp,48
  6e8: 00008067           ret
```



```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
PC→66c: 02113423        sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
 674: 03010413          add s0,sp,48
 678: 00050793          mv a5,a0
 67c: fcb43823          sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
 684: fe042423          sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
 68c: 0240006f          j 6b0 <main+0x48>
 690: fe842783          lw a5,-24(s0)
 694: 00078713          mv a4,a5
 698: fec42783          lw a5,-20(s0)
 69c: 00f707bb          addw a5,a4,a5
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```



```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
PC→66c: 02113423        sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
 674: 03010413          add s0,sp,48
 678: 00050793          mv a5,a0
 67c: fcb43823          sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
 684: fe042423          sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
 68c: 0240006f          j 6b0 <main+0x48>
 690: fe842783          lw a5,-24(s0)
 694: 00078713          mv a4,a5
 698: fec42783          lw a5,-20(s0)
 69c: 00f707bb          addw a5,a4,a5
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```



# Aufrufkonvention

Soviel wie möglich via Register

- Eingabedaten
- Rückgabewert

Gezielte Aufgaben

- a0-a7: Eingaben
- a0-a1: Ergebnisse
- ra: Rücksprungadresse

Caller-saved

- Aufrufer/Caller muss Werte selbst sichern, dürfen von Callee verändert werden

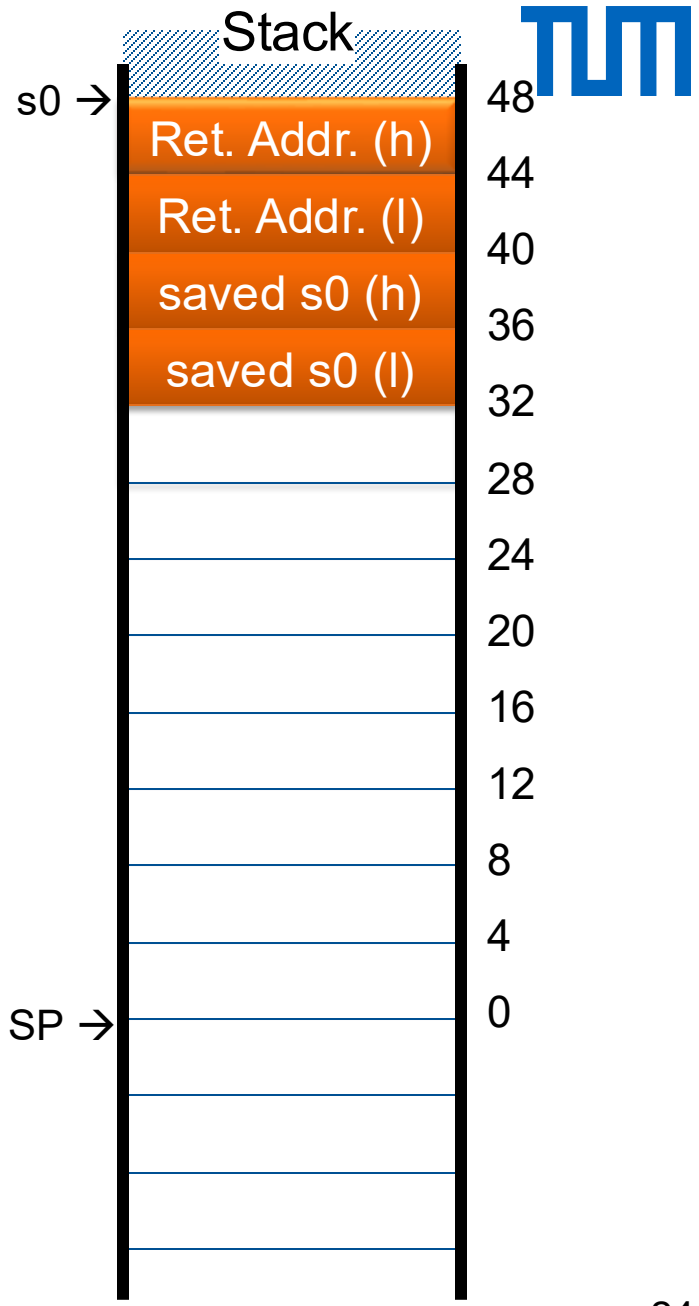
Callee-saved

- Aufgerufene Funktion darf Werte nicht verändern oder muss sie wieder herstellen

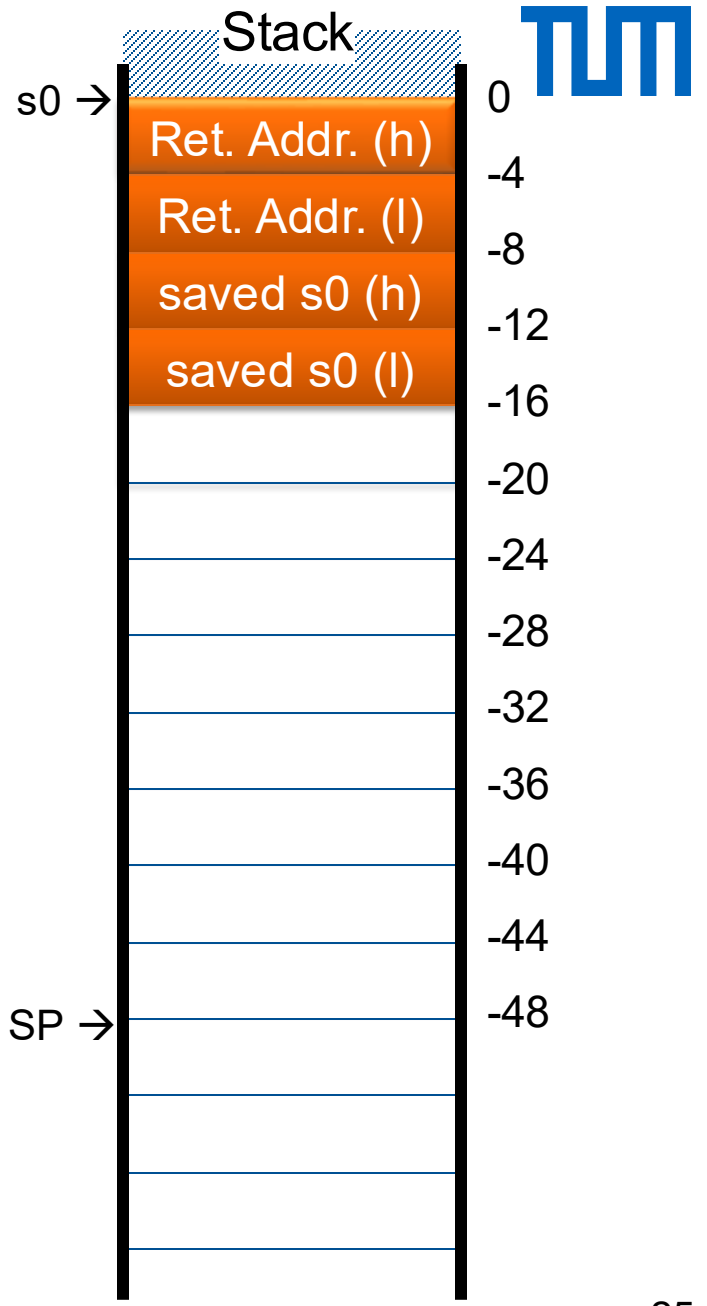
| Register | ABI Name | Description | Saver |
|---|---|---|---|
| x0 | zero | Hard-wired zero | — |
| x1 | ra | Return address | Caller |
| x2 | sp | Stack pointer | Callee |
| x3 | gp | Global pointer | — |
| x4 | tp | Thread pointer | — |
| x5–7 | t0–2 | Temporaries | Caller |
| x8 | s0/fp | Saved register/frame pointer | Callee |
| x9 | s1 | Saved register | Callee |
| x10–11 | a0–1 | Function arguments/return values | Caller |
| x12–17 | a2–7 | Function arguments | Caller |
| x18–27 | s2–11 | Saved registers | Callee |
| x28–31 | t3–6 | Temporaries | Caller |
| f0–7 | ft0–7 | FP temporaries | Caller |
| f8–9 | fs0–1 | FP saved registers | Callee |
| f10–11 | fa0–1 | FP arguments/return values | Caller |
| f12–17 | fa2–7 | FP arguments | Caller |
| f18–27 | fs2–11 | FP saved registers | Callee |
| f28–31 | ft8–11 | FP temporaries | Caller |

From: https://riscv.org/technical/specifications/

```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
PC→674: 03010413        add s0,sp,48
 678: 00050793          mv a5,a0
 67c: fcb43823          sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
 684: fe042423          sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
 68c: 0240006f          j 6b0 <main+0x48>
 690: fe842783          lw a5,-24(s0)
 694: 00078713          mv a4,a5
 698: fec42783          lw a5,-20(s0)
 69c: 00f707bb          addw a5,a4,a5
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```

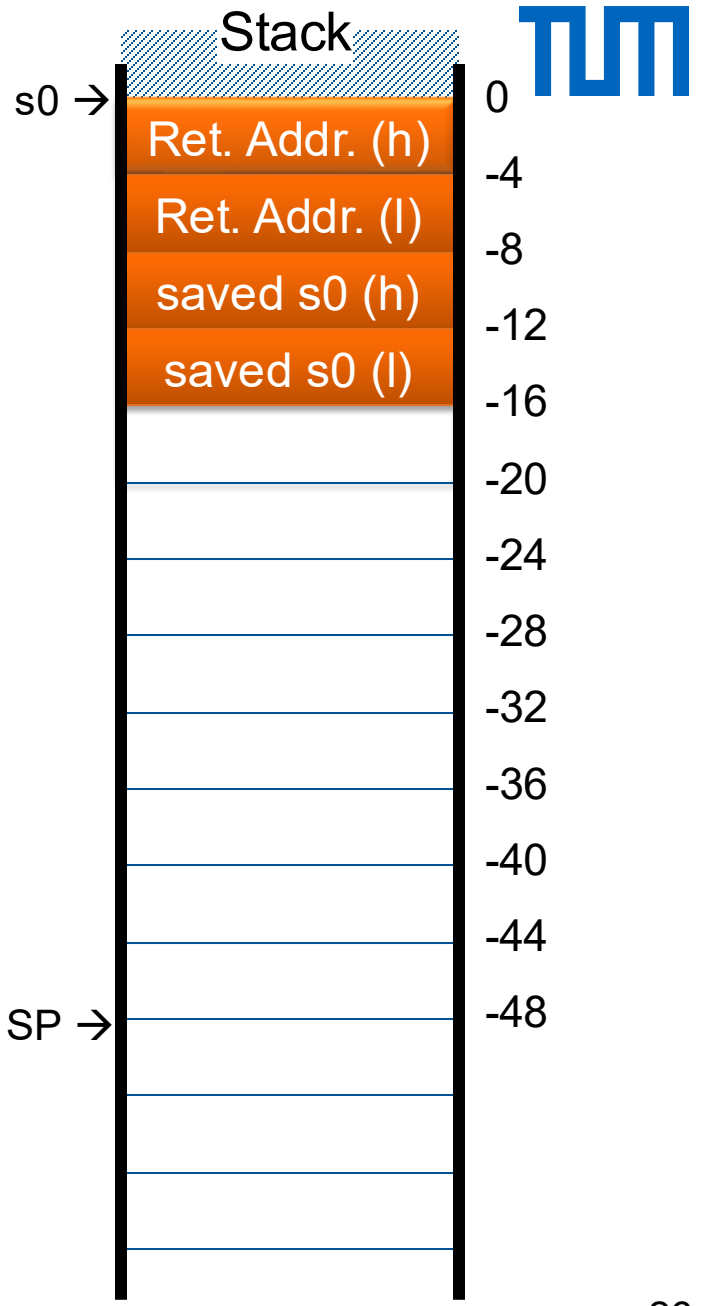


```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
PC→674: 03010413        add s0,sp,48
 678: 00050793          mv a5,a0
 67c: fcb43823          sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
 684: fe042423          sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
 68c: 0240006f          j 6b0 <main+0x48>
 690: fe842783          lw a5,-24(s0)
 694: 00078713          mv a4,a5
 698: fec42783          lw a5,-20(s0)
 69c: 00f707bb          addw a5,a4,a5
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```



```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
 674: 03010413          add s0,sp,48
PC→678: 00050793        mv a5,a0
 67c: fcb43823          sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
 684: fe042423          sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
 68c: 0240006f          j 6b0 <main+0x48>
 690: fe842783          lw a5,-24(s0)
 694: 00078713          mv a4,a5
 698: fec42783          lw a5,-20(s0)
 69c: 00f707bb          addw a5,a4,a5
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```

| a0 | Input Value |
|---|---|
| a1 | |
| a2 | |
| a3 | |
| a4 | |
| a5 | |
| a6 | |



# Aufrufkonvention

Soviel wie möglich via Register

- Eingabedaten
- Rückgabewert

Gezielte Aufgaben

- a0-a7: Eingaben
- a0-a1: Ergebnisse
- ra: Rücksprungadresse

Caller-saved

- Aufrufer/Caller muss Werte selbst sichern, dürfen von Callee verändert werden

Callee-saved

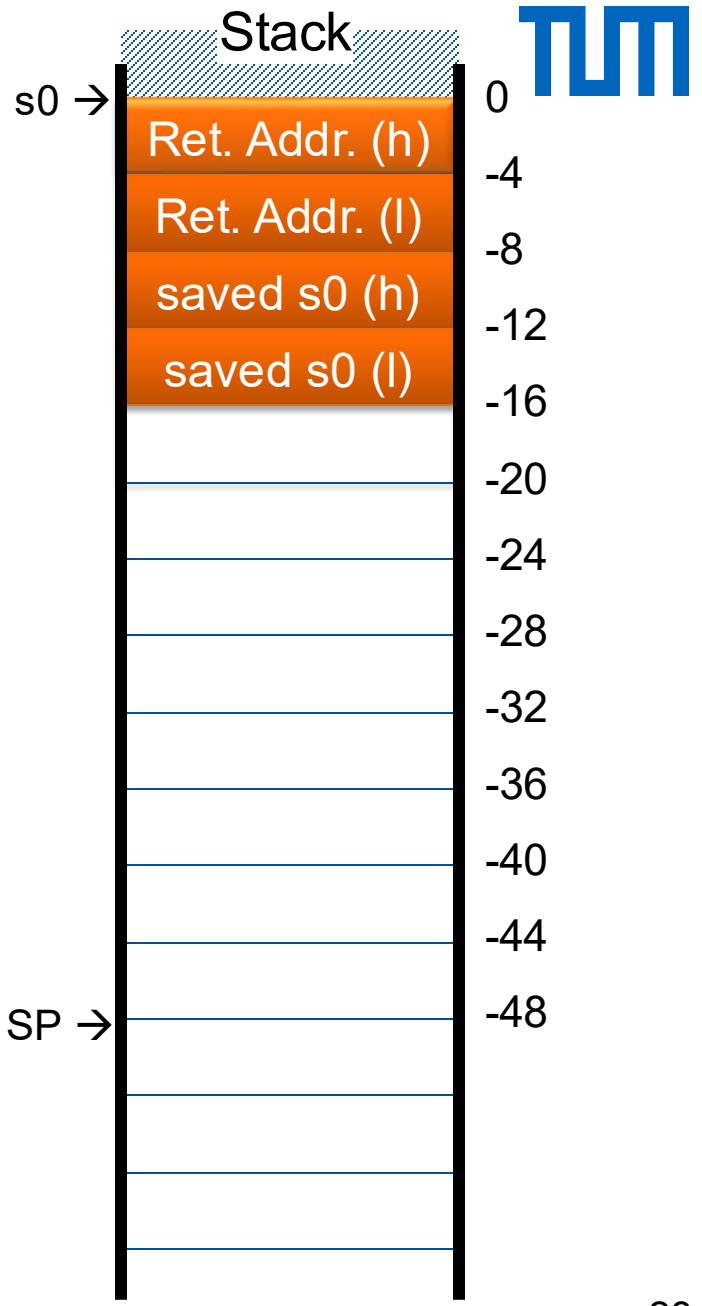
- Aufgerufene Funktion darf Werte nicht verändern oder muss sie wieder herstellen

| Register | ABI Name | Description | Saver |
|---|---|---|---|
| x0 | zero | Hard-wired zero | — |
| x1 | ra | Return address | Caller |
| x2 | sp | Stack pointer | Callee |
| x3 | gp | Global pointer | — |
| x4 | tp | Thread pointer | — |
| x5–7 | t0–2 | Temporaries | Caller |
| x8 | s0/fp | Saved register/frame pointer | Callee |
| x9 | s1 | Saved register | Callee |
| x10–11 | a0–1 | Function arguments/return values | Caller |
| x12–17 | a2–7 | Function arguments | Caller |
| x18–27 | s2–11 | Saved registers | Callee |
| x28–31 | t3–6 | Temporaries | Caller |
| f0–7 | ft0–7 | FP temporaries | Caller |
| f8–9 | fs0–1 | FP saved registers | Callee |
| f10–11 | fa0–1 | FP arguments/return values | Caller |
| f12–17 | fa2–7 | FP arguments | Caller |
| f18–27 | fs2–11 | FP saved registers | Callee |
| f28–31 | ft8–11 | FP temporaries | Caller |

From: https://riscv.org/technical/specifications/

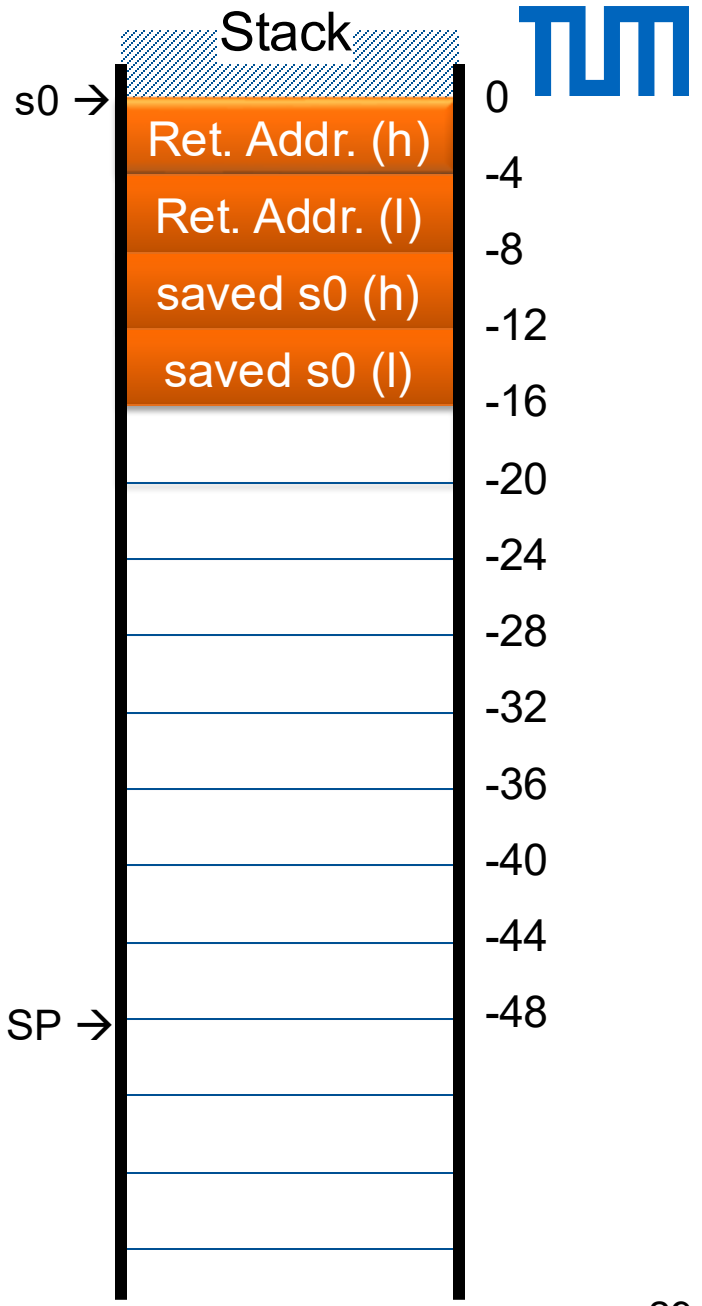
```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
 674: 03010413          add s0,sp,48
PC→678: 00050793        mv a5,a0
 67c: fcb43823          sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
 684: fe042423          sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
 68c: 0240006f          j 6b0 <main+0x48>
 690: fe842783          lw a5,-24(s0)
 694: 00078713          mv a4,a5
 698: fec42783          lw a5,-20(s0)
 69c: 00f707bb          addw a5,a4,a5
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```

| | |
|---|---|
| a0 | Eing. Param. |
| a1 | |
| a2 | |
| a3 | |
| a4 | |
| a5 | Eing. Param. |
| a6 | |



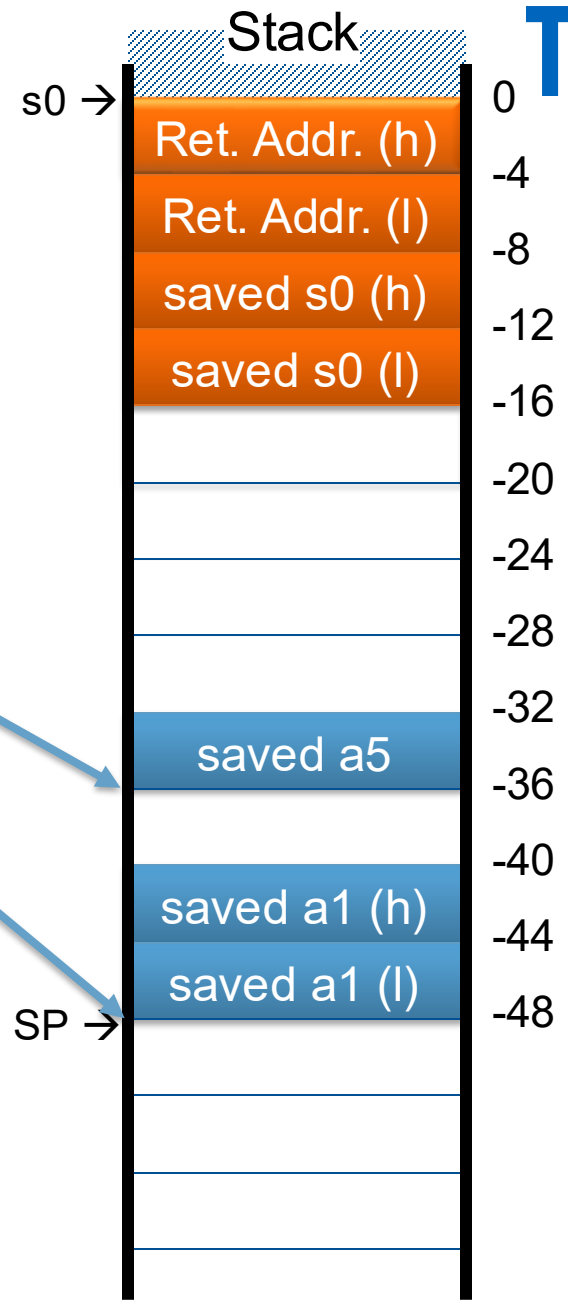
```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
 674: 03010413          add s0,sp,48
 678: 00050793          mv a5,a0
PC→67c: fcb43823        sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
 684: fe042423          sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
 68c: 0240006f          j 6b0 <main+0x48>
 690: fe842783          lw a5,-24(s0)
 694: 00078713          mv a4,a5
 698: fec42783          lw a5,-20(s0)
 69c: 00f707bb          addw a5,a4,a5
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```

| | |
|---|---|
| a0 | Eing. Param. |
| a1 | |
| a2 | |
| a3 | |
| a4 | |
| a5 | Eing. Param. |
| a6 | |



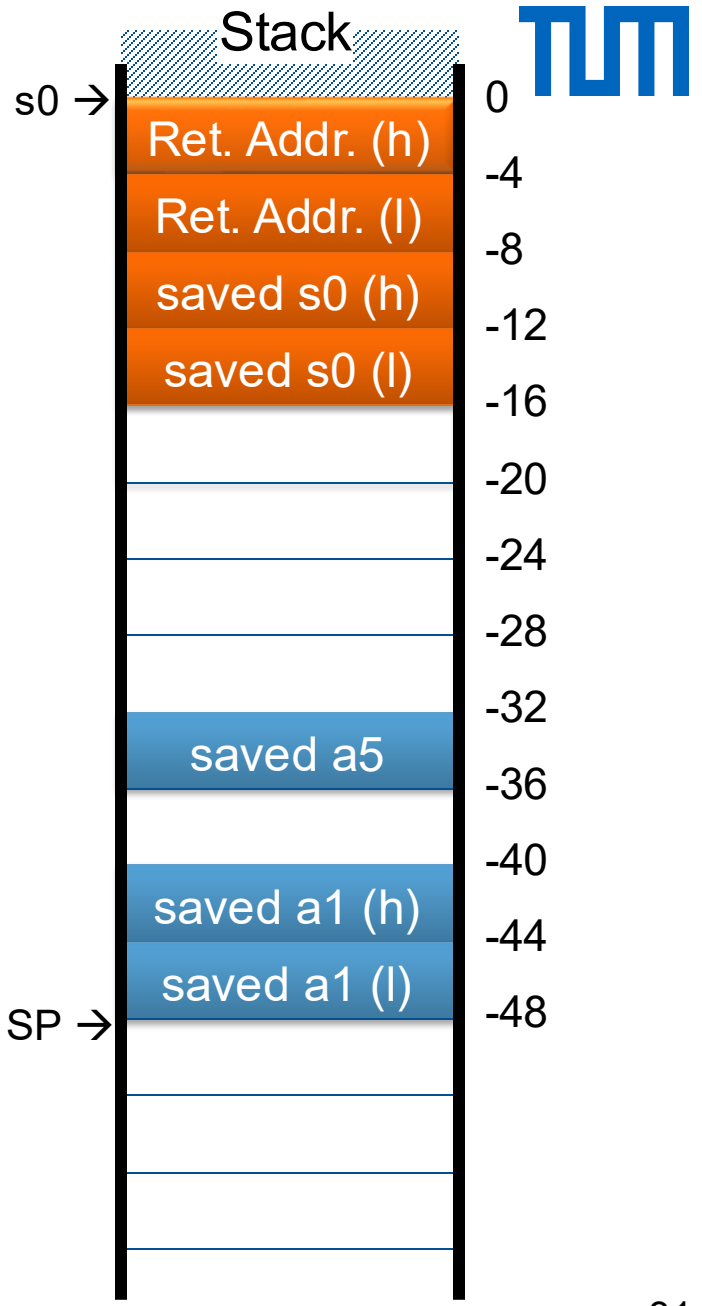
```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
 674: 03010413          add s0,sp,48
 678: 00050793          mv a5,a0
PC→67c: fcb43823        sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
 684: fe042423          sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
 68c: 0240006f          j 6b0 <main+0x48>
 690: fe842783          lw a5,-24(s0)
 694: 00078713          mv a4,a5
 698: fec42783          lw a5,-20(s0)
 69c: 00f707bb          addw a5,a4,a5
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```

| | |
|---|---|
| a0 | Eing. Param. |
| a1 | |
| a2 | |
| a3 | |
| a4 | |
| a5 | Eing. Param. |
| a6 | |

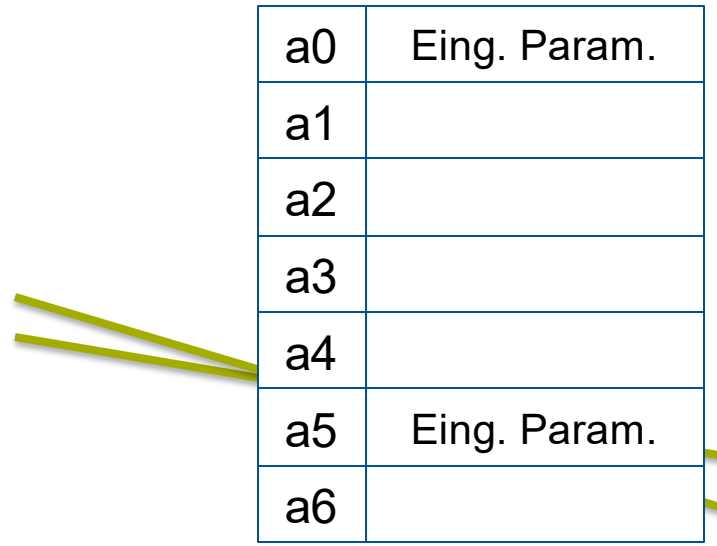


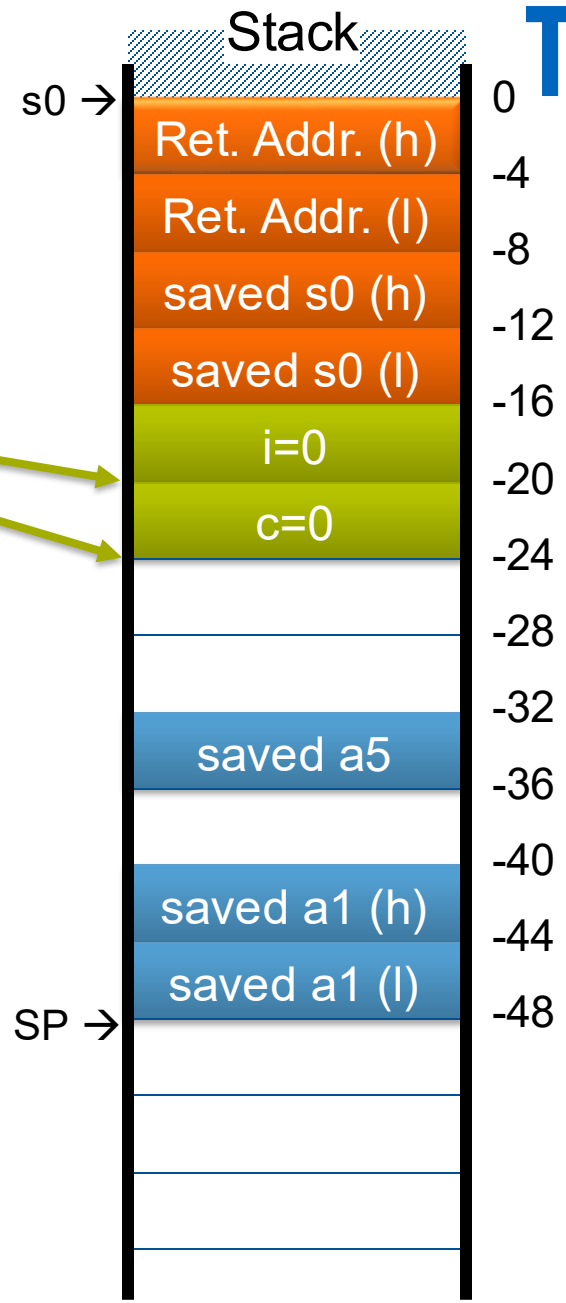
```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
 674: 03010413          add s0,sp,48
 678: 00050793          mv a5,a0
 67c: fcb43823          sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
PC→684: fe042423        sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
 68c: 0240006f          j 6b0 <main+0x48>
 690: fe842783          lw a5,-24(s0)
 694: 00078713          mv a4,a5
 698: fec42783          lw a5,-20(s0)
 69c: 00f707bb          addw a5,a4,a5
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```

| | |
|---|---|
| a0 | Eing. Param. |
| a1 | |
| a2 | |
| a3 | |
| a4 | |
| a5 | Eing. Param. |
| a6 | |



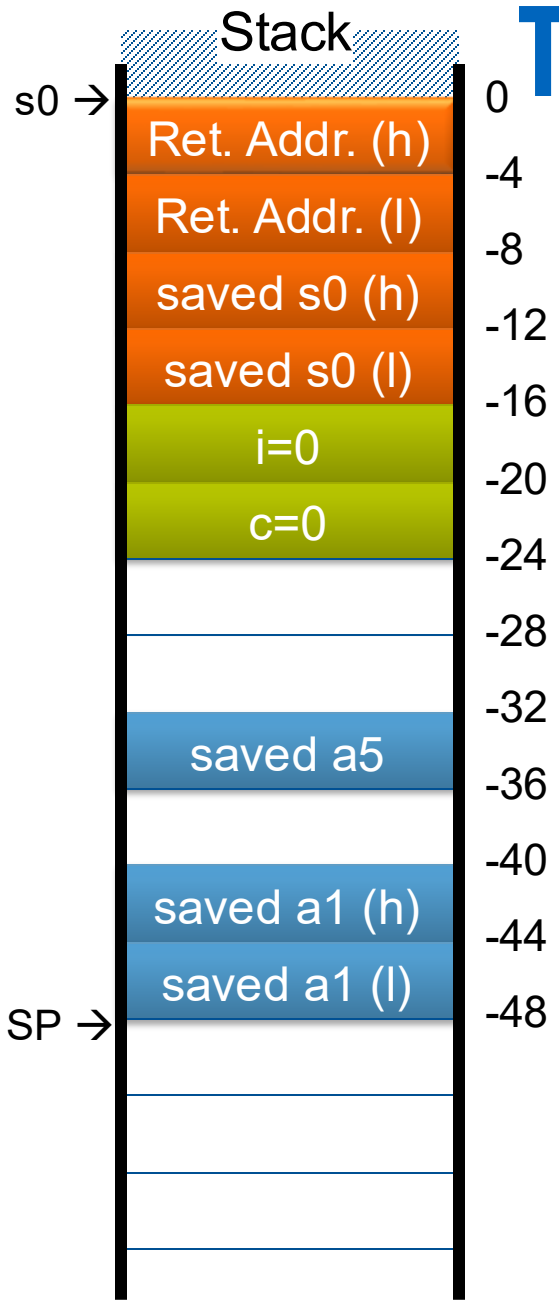
```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
 674: 03010413          add s0,sp,48
 678: 00050793          mv a5,a0
 67c: fcb43823          sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
PC→684: fe042423        sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
 68c: 0240006f          j 6b0 <main+0x48>
 690: fe842783          lw a5,-24(s0)
 694: 00078713          mv a4,a5
 698: fec42783          lw a5,-20(s0)
 69c: 00f707bb          addw a5,a4,a5
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```



```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
 674: 03010413          add s0,sp,48
 678: 00050793          mv a5,a0
 67c: fcb43823          sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
 684: fe042423          sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
PC→68c: 0240006f        j 6b0 <main+0x48>
 690: fe842783          lw a5,-24(s0)
 694: 00078713          mv a4,a5
 698: fec42783          lw a5,-20(s0)
 69c: 00f707bb          addw a5,a4,a5
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```

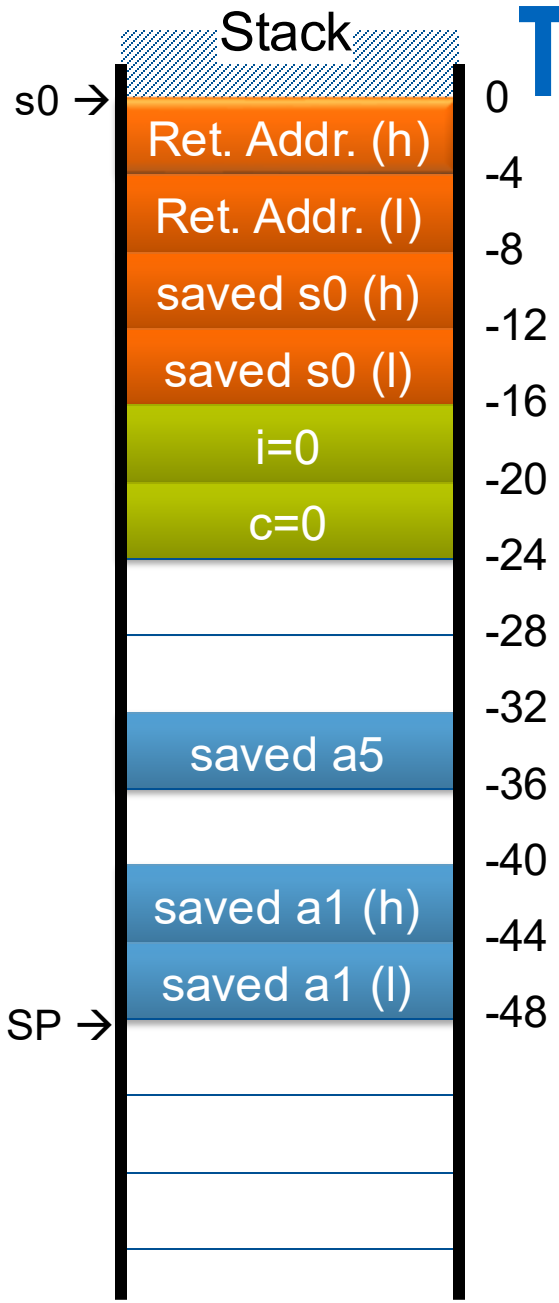
| | |
|---|---|
| a0 | Eing. Param. |
| a1 | |
| a2 | |
| a3 | |
| a4 | |
| a5 | Eing. Param. |
| a6 | |



```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
 674: 03010413          add s0,sp,48
 678: 00050793          mv a5,a0
 67c: fcb43823          sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
 684: fe042423          sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
PC→68c: 0240006f        j 6b0 <main+0x48>
 690: fe842783          lw a5,-24(s0)
 694: 00078713          mv a4,a5
 698: fec42783          lw a5,-20(s0)
 69c: 00f707bb          addw a5,a4,a5
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```

Initialization Complete
Start Loop

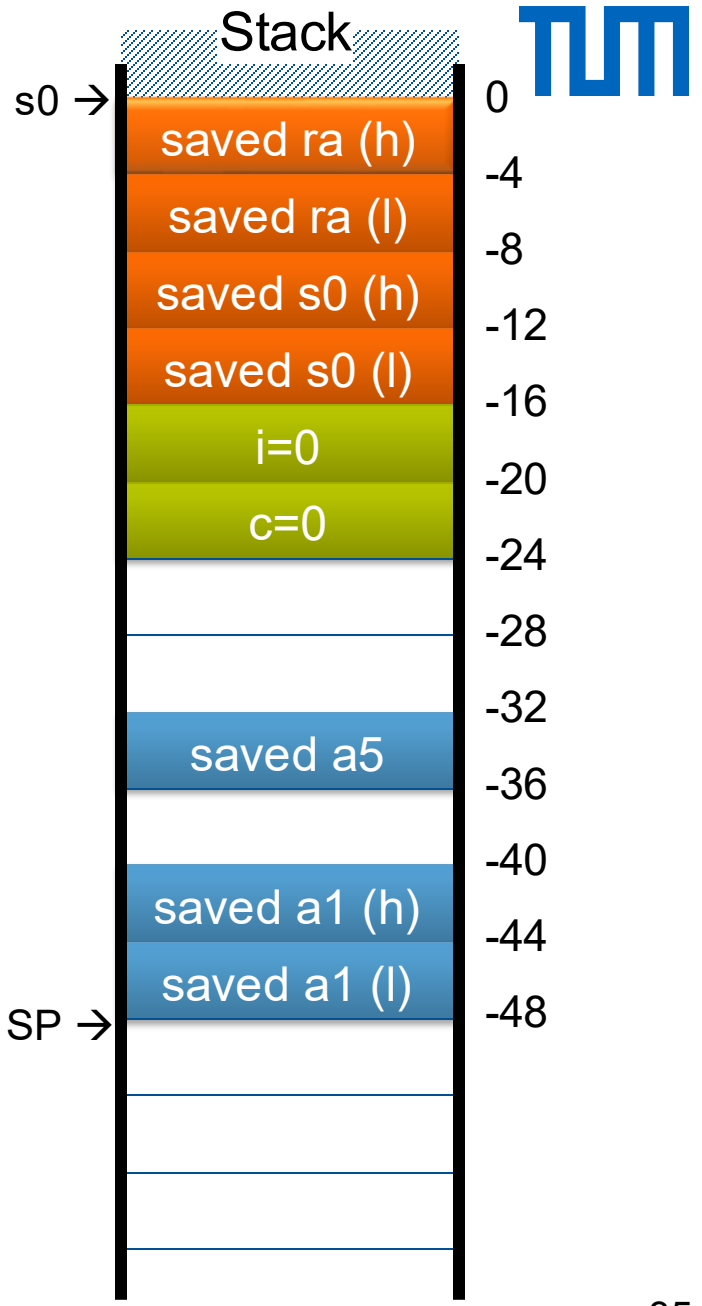
| | |
|---|---|
| | ram. |
| a1 | |
| a2 | |
| a3 | |
| a4 | |
| a5 | Eing. Param. |
| a6 | |



```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
 674: 03010413          add s0,sp,48
 678: 00050793          mv a5,a0
 67c: fcb43823          sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
 684: fe042423          sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
PC→68c: 0240006f        j 6b0 <main+0x48>
 690: fe842783          lw a5,-24(s0)
 694: 00078713          mv a4,a5
 698: fec42783          lw a5,-20(s0)
 69c: 00f707bb          addw a5,a4,a5
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```

Skip loop body

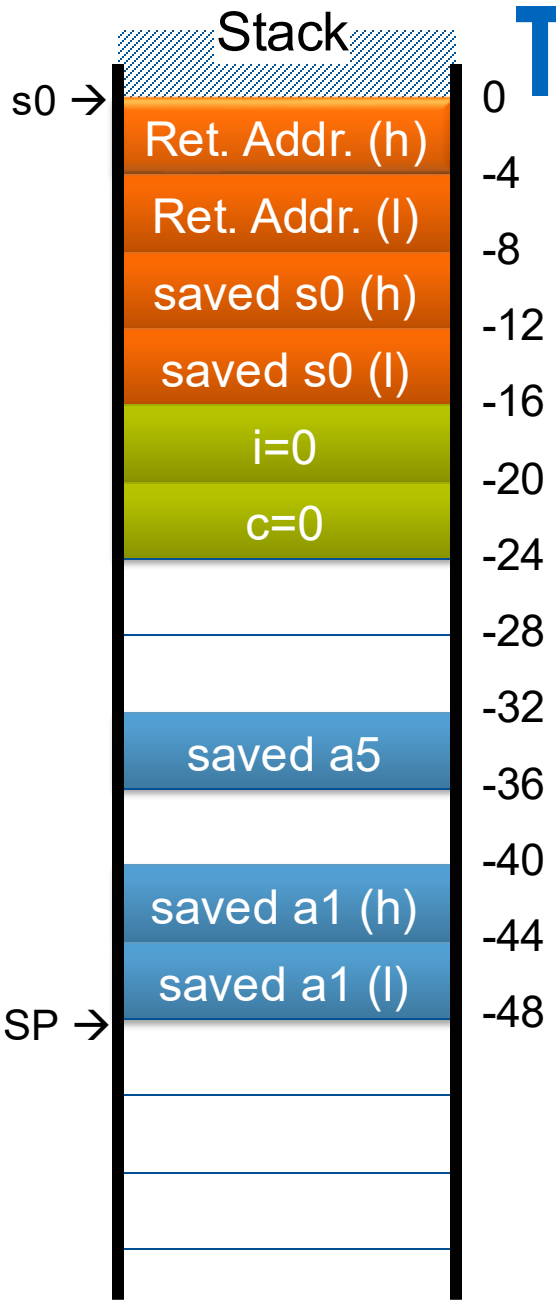
| | |
|---|---|
| a0 | Eing. Param. |
| a1 | |
| a2 | |
| a3 | |
| a4 | |
| a5 | Eing. Param. |
| a6 | |



```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
 674: 03010413          add s0,sp,48
 678: 00050793          mv a5,a0
 67c: fcb43823          sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
 684: fe042423          sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
 68c: 0240006f          j 6b0 <main+0x48>
 690: fe842783          lw a5,-24(s0)
 694: 00078713          mv a4,a5
 698: fec42783          lw a5,-20(s0)
 69c: 00f707bb          addw a5,a4,a5
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
PC→6b0: fec42783        lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```

| | |
|---|---|
| a0 | Eing. Param. |
| a1 | |
| a2 | |
| a3 | |
| a4 | |
| a5 | Eing. Param. |
| a6 | |

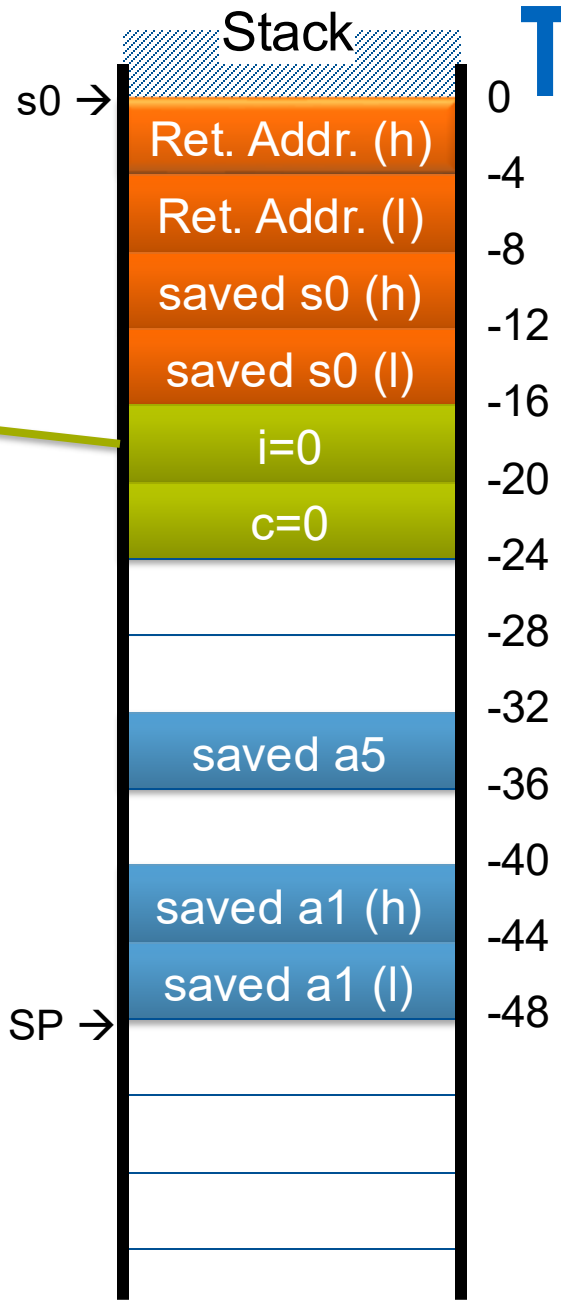
Check Loop Conditions



```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
 674: 03010413          add s0,sp,48
 678: 00050793          mv a5,a0
 67c: fcb43823          sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
 684: fe042423          sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
 68c: 0240006f          j 6b0 <main+0x48>
 690: fe842783          lw a5,-24(s0)
 694: 00078713          mv a4,a5
 698: fec42783          lw a5,-20(s0)
 69c: 00f707bb          addw a5,a4,a5
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
PC→6b0: fec42783        lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```

| | |
|---|---|
| a0 | Eing. Param. |
| a1 | |
| a2 | |
| a3 | |
| a4 | |
| a5 | Variable i=0 |
| a6 | |

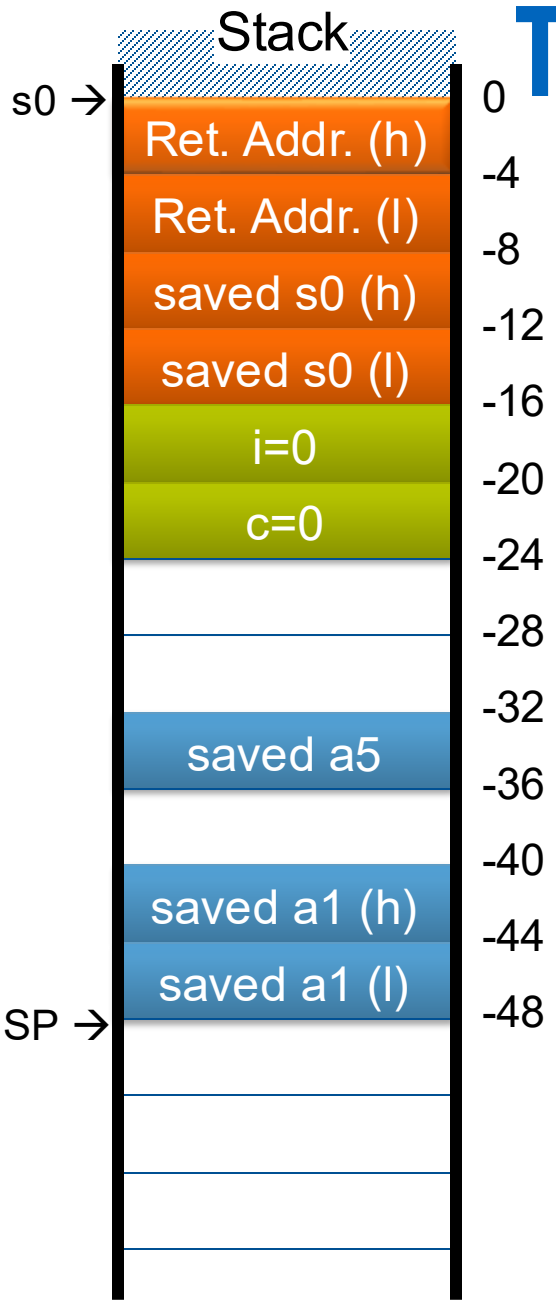
Check Loop Conditions



```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
 674: 03010413          add s0,sp,48
 678: 00050793          mv a5,a0
 67c: fcb43823          sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
 684: fe042423          sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
 68c: 0240006f          j 6b0 <main+0x48>
 690: fe842783          lw a5,-24(s0)
 694: 00078713          mv a4,a5
 698: fec42783          lw a5,-20(s0)
 69c: 00f707bb          addw a5,a4,a5
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
PC→6b4: 0007871b        sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```

| | |
|---|---|
| a0 | Eing. Param. |
| a1 | |
| a2 | |
| a3 | |
| a4 | |
| a5 | Variable i=0 |
| a6 | |

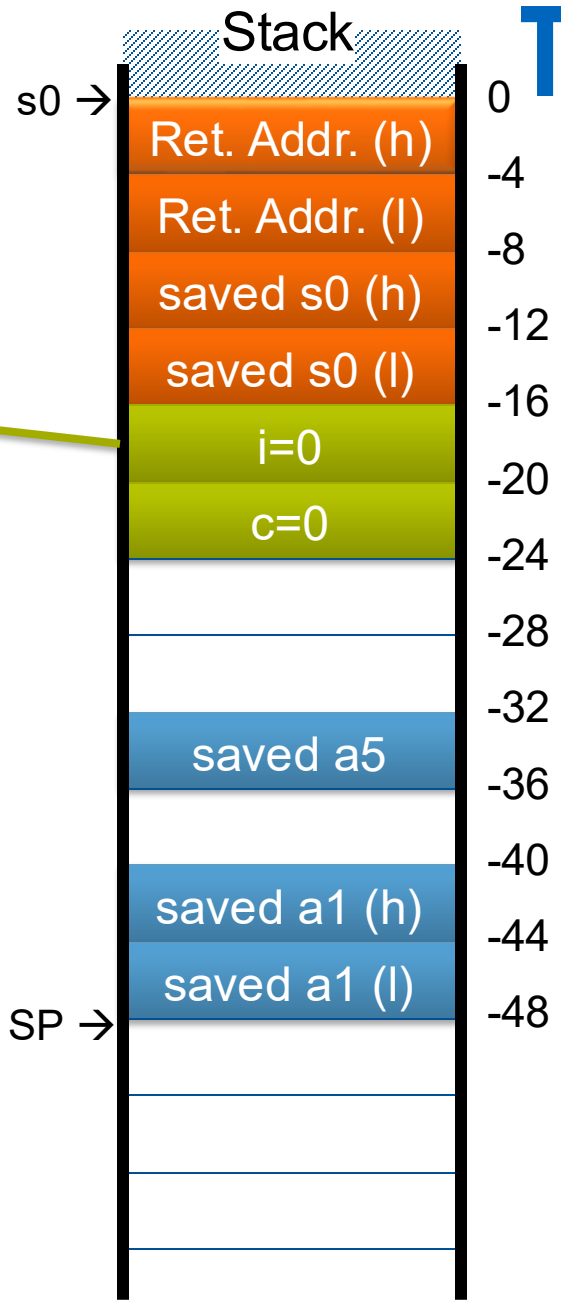
sext.w a4,a5 = addiw a4,ra5,0
Sign extends a5 into a4



```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
 674: 03010413          add s0,sp,48
 678: 00050793          mv a5,a0
 67c: fcb43823          sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
 684: fe042423          sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
 68c: 0240006f          j 6b0 <main+0x48>
 690: fe842783          lw a5,-24(s0)
 694: 00078713          mv a4,a5
 698: fec42783          lw a5,-20(s0)
 69c: 00f707bb          addw a5,a4,a5
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
PC→6b4: 0007871b        sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```

| | |
|---|---|
| a0 | Eing. Param. |
| a1 | |
| a2 | |
| a3 | |
| a4 | sign ext. i=0 |
| a5 | Variable I=0 |
| a6 | |

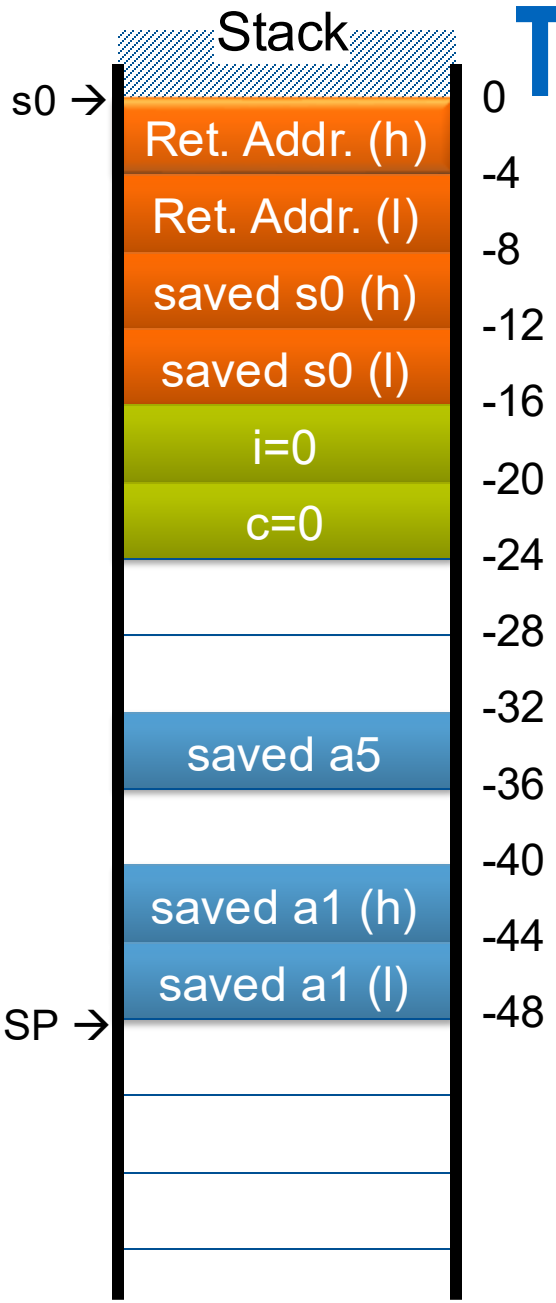
sext.w a4,a5 = addiw a4,ra5,0
Sign extends a5 into a4



```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
 674: 03010413          add s0,sp,48
 678: 00050793          mv a5,a0
 67c: fcb43823          sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
 684: fe042423          sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
 68c: 0240006f          j 6b0 <main+0x48>
 690: fe842783          lw a5,-24(s0)
 694: 00078713          mv a4,a5
 698: fec42783          lw a5,-20(s0)
 69c: 00f707bb          addw a5,a4,a5
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
PC→6b8: 00900793        li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```

| | |
|---|---|
| a0 | Eing. Param. |
| a1 | |
| a2 | |
| a3 | |
| a4 | sign ext. i=0 |
| a5 | Variable i=0 |
| a6 | |

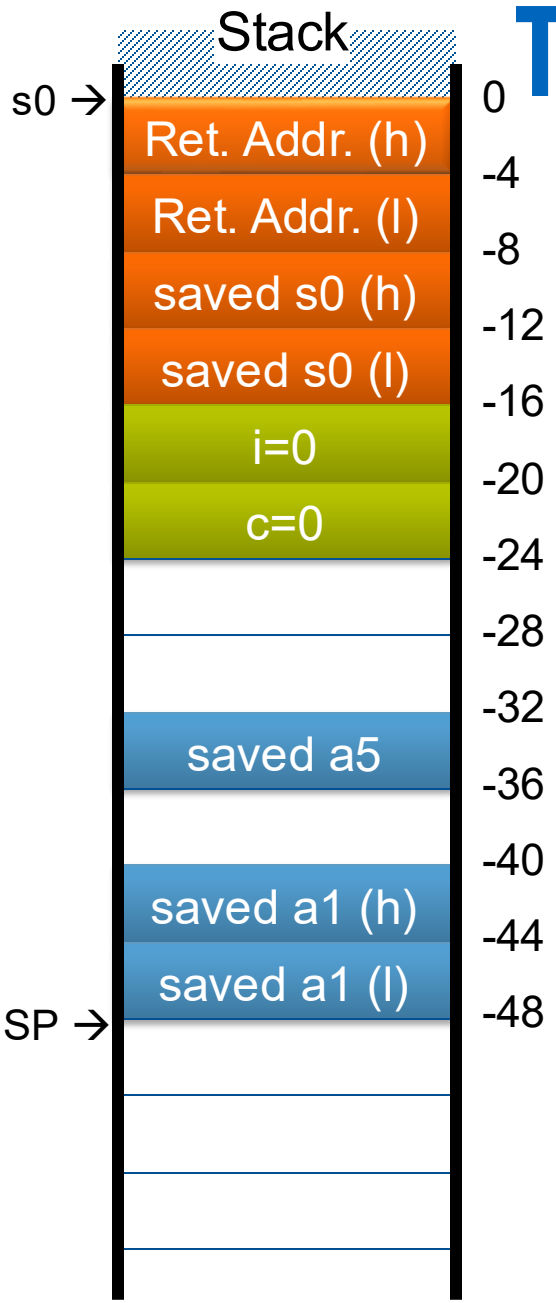
li a5,9 = addi a5,zero,9



```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
 674: 03010413          add s0,sp,48
 678: 00050793          mv a5,a0
 67c: fcb43823          sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
 684: fe042423          sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
 68c: 0240006f          j 6b0 <main+0x48>
 690: fe842783          lw a5,-24(s0)
 694: 00078713          mv a4,a5
 698: fec42783          lw a5,-20(s0)
 69c: 00f707bb          addw a5,a4,a5
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
PC→6b8: 00900793        li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```

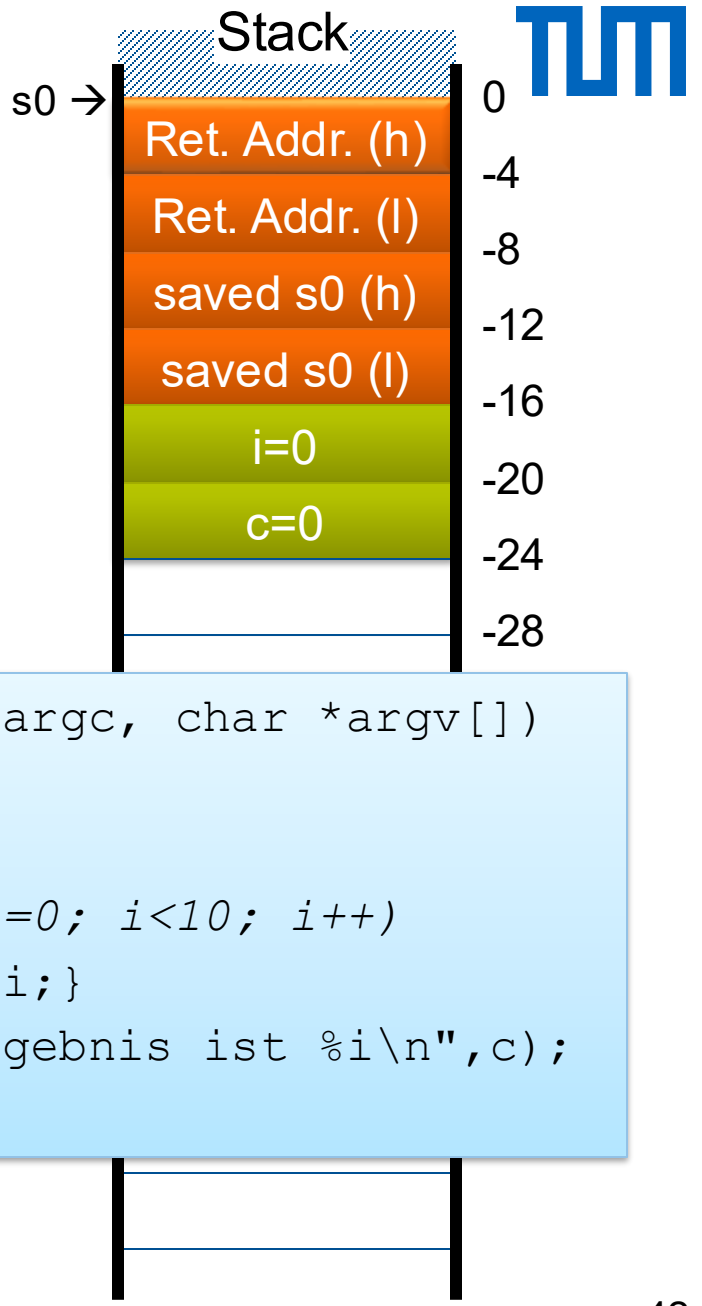
| | |
|---|---|
| a0 | Eing. Param. |
| a1 | |
| a2 | |
| a3 | |
| a4 | sign ext. i=0 |
| a5 | Wert 9 |
| a6 | |

li a5,9 = addi a5,zero,9



```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
 674: 03010413          add s0,sp,48
 678: 00050793          mv a5,a0
 67c: fcb43823          sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
 684: fe042423          sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
 68c: 0240006f          j 6b0 <main+0x48>
 690: fe842783          lw a5,-24(s0)
 694: 00078713          mv a4,a5
 698: fec42783          lw a5,-20(s0)
 69c: 00f707bb          addw a5,a4,a5
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
PC→6bc: fce7dae3        bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```

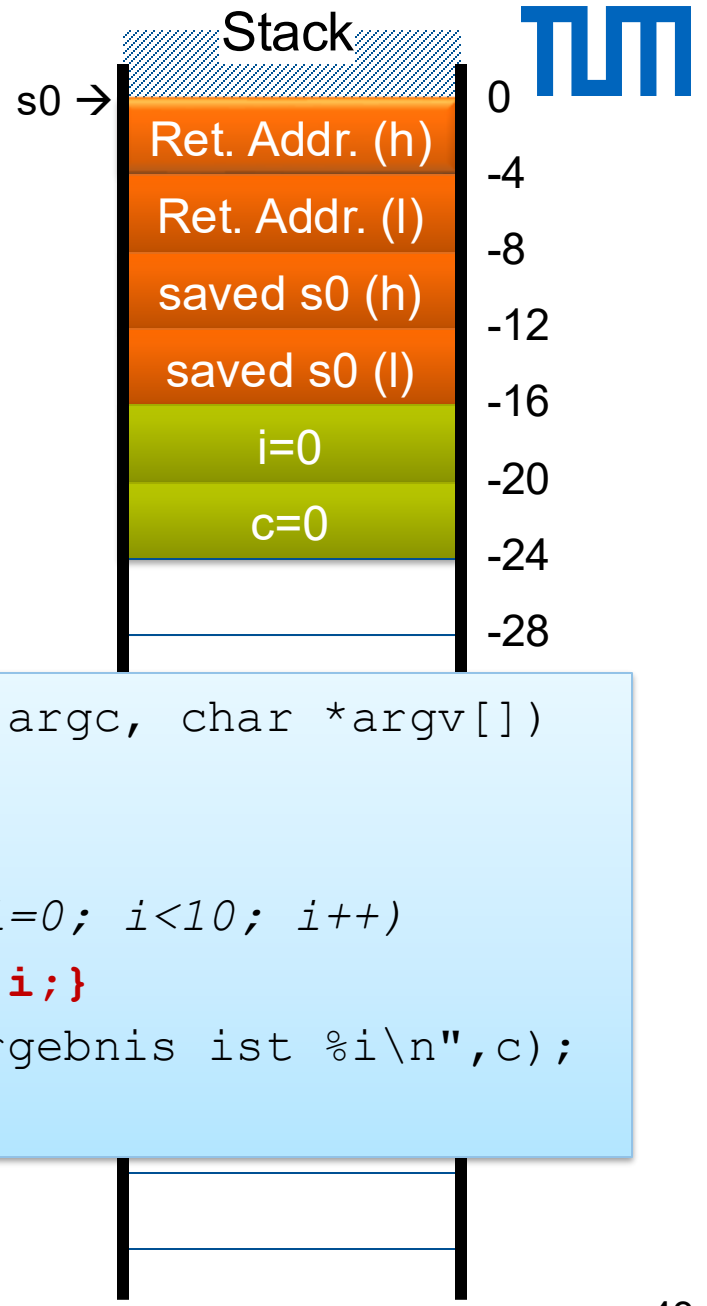
| | |
|---|---|
| a0 | Input Value |
| a1 | |
| a2 | |
| a3 | |
| a4 | sign ext. i=0 |
| a5 | Wert 9 |
| a6 | |




```
int main(int argc, char *argv[])
{
    int c=0;
    for (int i=0; i<10; i++)
    { c = c + i;}
    printf("Ergebnis ist %i\n",c);
}
```

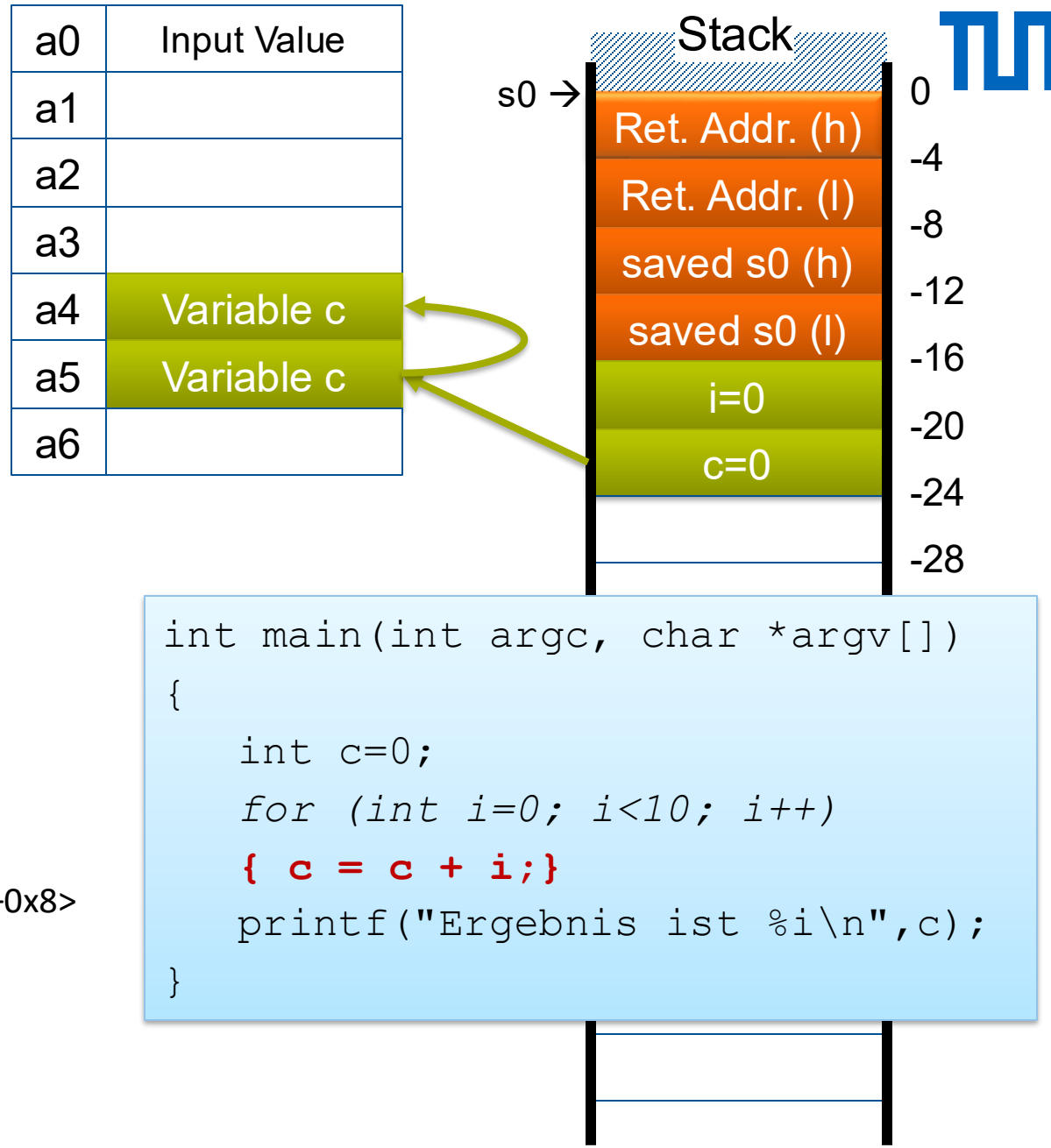
```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
 674: 03010413          add s0,sp,48
 678: 00050793          mv a5,a0
 67c: fcb43823          sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
 684: fe042423          sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
 68c: 0240006f          j 6b0 <main+0x48>
PC→690: fe842783        lw a5,-24(s0)
 694: 00078713          mv a4,a5
 698: fec42783          lw a5,-20(s0)
 69c: 00f707bb          addw a5,a4,a5
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```

| | |
|---|---|
| a0 | Input Value |
| a1 | |
| a2 | |
| a3 | |
| a4 | sign ext. i=0 |
| a5 | Wert 9 |
| a6 | |



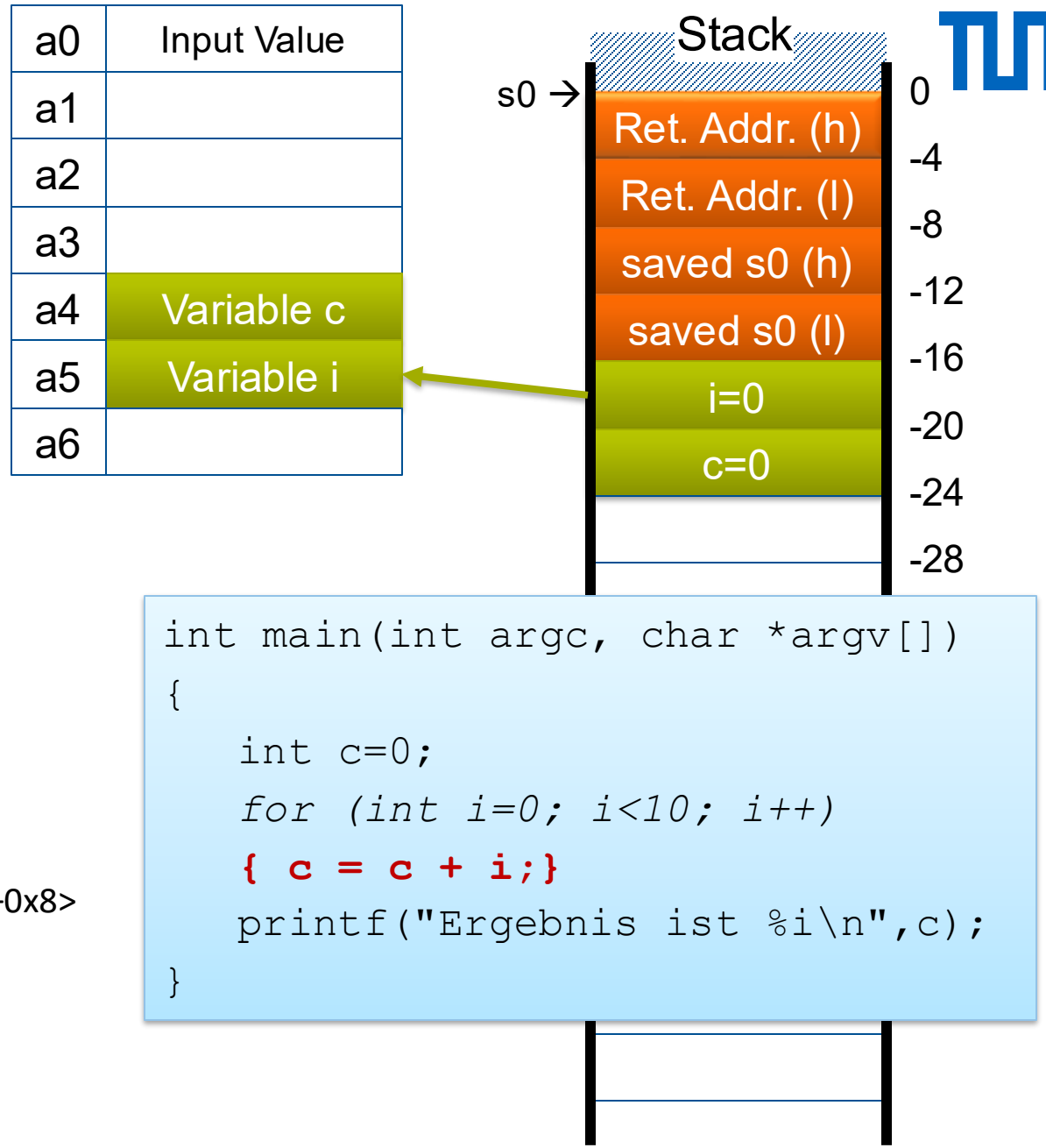

```
int main(int argc, char *argv[])
{
    int c=0;
    for (int i=0; i<10; i++)
    { c = c + i;}
    printf("Ergebnis ist %i\n",c);
}
```

```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
 674: 03010413          add s0,sp,48
 678: 00050793          mv a5,a0
 67c: fcb43823          sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
 684: fe042423          sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
 68c: 0240006f          j 6b0 <main+0x48>
PC→690: fe842783        lw a5,-24(s0)
 694: 00078713          mv a4,a5
 698: fec42783          lw a5,-20(s0)
 69c: 00f707bb          addw a5,a4,a5
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```




```
int main(int argc, char *argv[])
{
    int c=0;
    for (int i=0; i<10; i++)
    { c = c + i;}
    printf("Ergebnis ist %i\n",c);
}
```

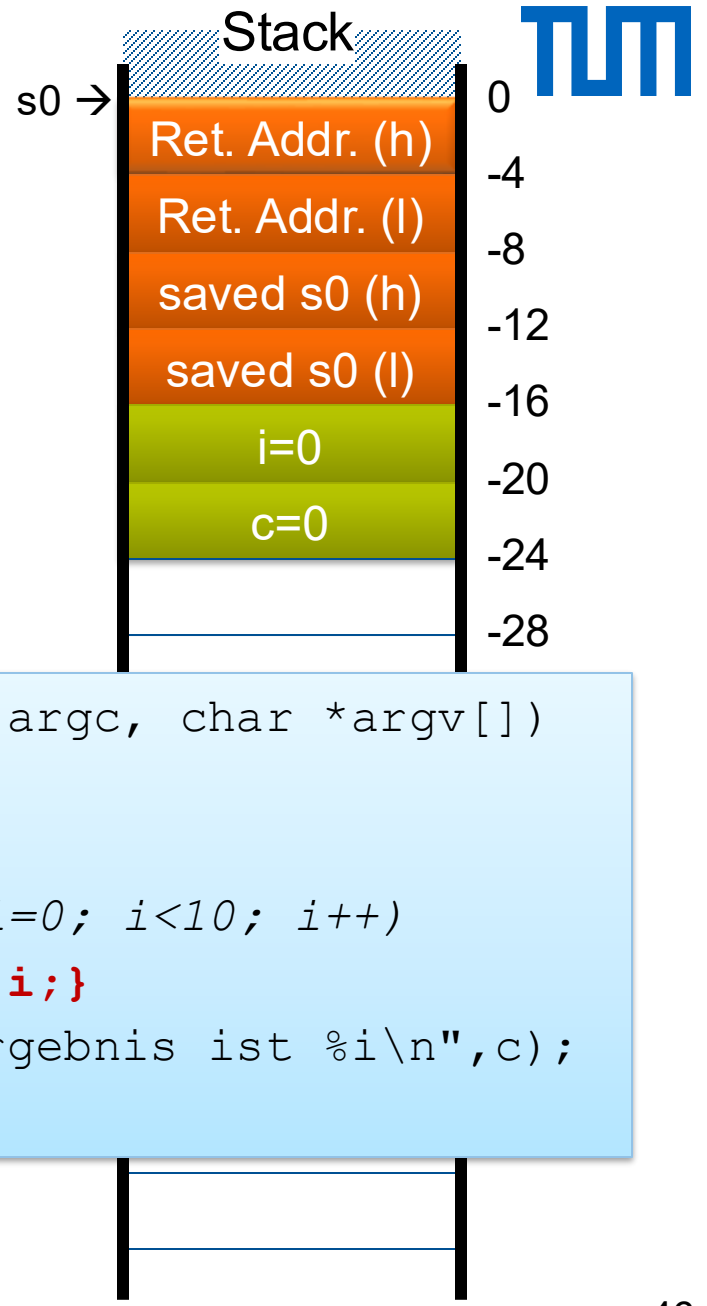
```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
 674: 03010413          add s0,sp,48
 678: 00050793          mv a5,a0
 67c: fcb43823          sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
 684: fe042423          sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
 68c: 0240006f          j 6b0 <main+0x48>
 690: fe842783          lw a5,-24(s0)
 694: 00078713          mv a4,a5
PC→698: fec42783        lw a5,-20(s0)
 69c: 00f707bb          addw a5,a4,a5
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```




```
int main(int argc, char *argv[])
{
    int c=0;
    for (int i=0; i<10; i++)
    { c = c + i;}
    printf("Ergebnis ist %i\n",c);
}
```

```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
 674: 03010413          add s0,sp,48
 678: 00050793          mv a5,a0
 67c: fcb43823          sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
 684: fe042423          sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
 68c: 0240006f          j 6b0 <main+0x48>
 690: fe842783          lw a5,-24(s0)
 694: 00078713          mv a4,a5
 698: fec42783          lw a5,-20(s0)
PC→69c: 00f707bb        addw a5,a4,a5
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```

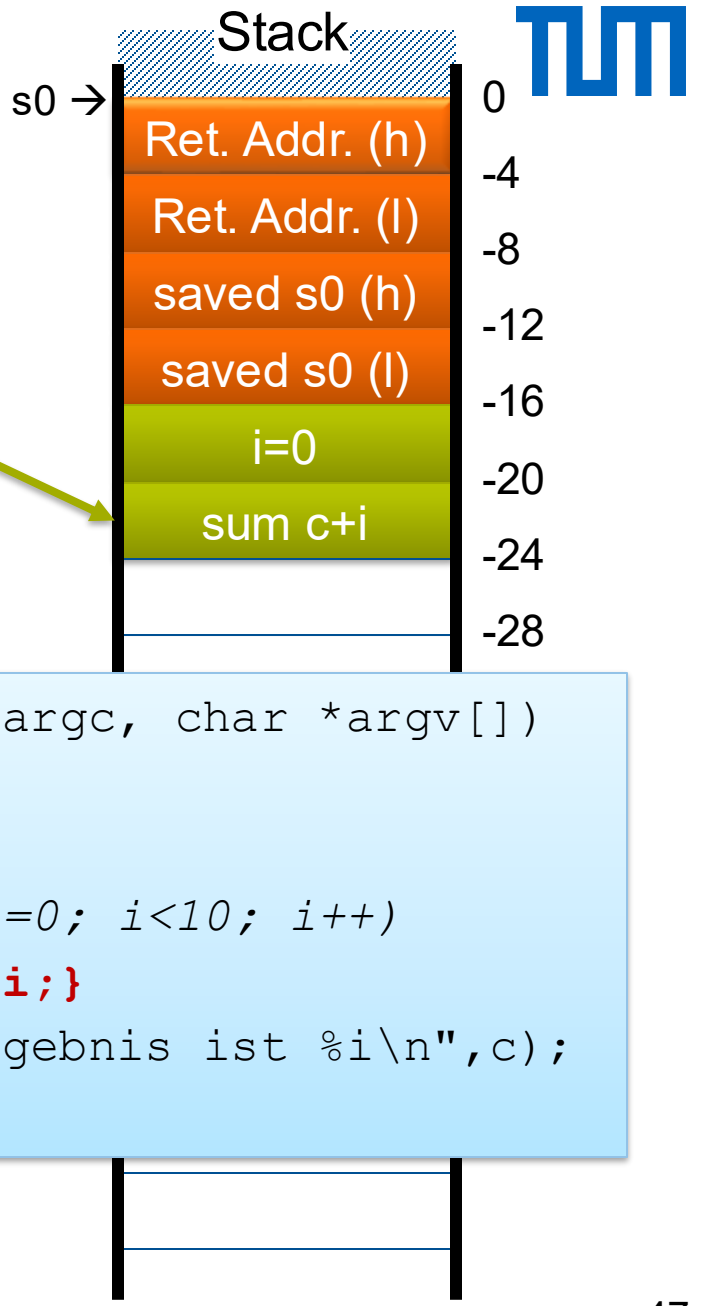
| | |
|---|---|
| a0 | Input Value |
| a1 | |
| a2 | |
| a3 | |
| a4 | Variable c |
| a5 | sum c+i |
| a6 | |




```
int main(int argc, char *argv[])
{
    int c=0;
    for (int i=0; i<10; i++)
    { c = c + i;}
    printf("Ergebnis ist %i\n",c);
}
```

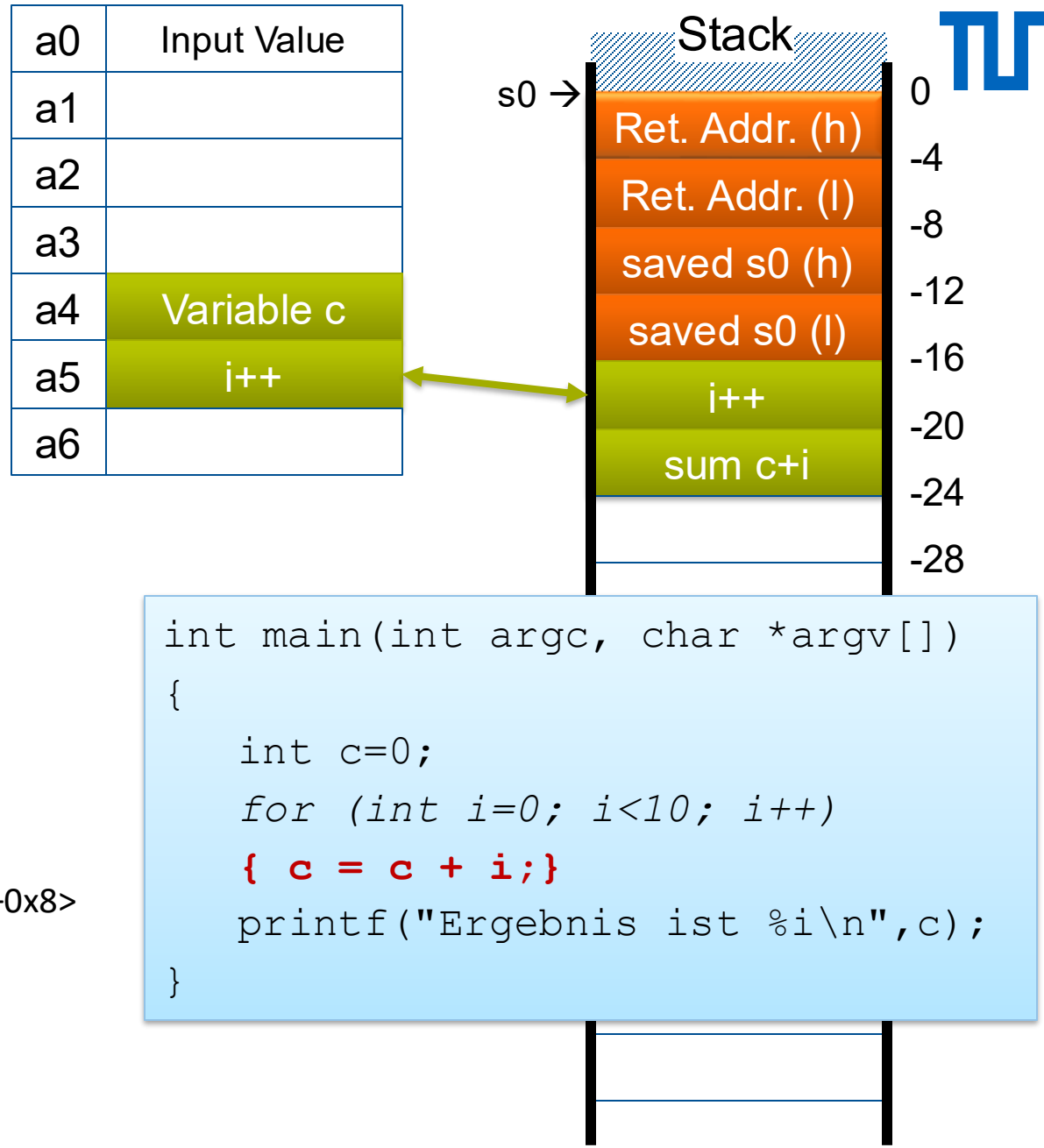
```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
 674: 03010413          add s0,sp,48
 678: 00050793          mv a5,a0
 67c: fcb43823          sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
 684: fe042423          sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
 68c: 0240006f          j 6b0 <main+0x48>
 690: fe842783          lw a5,-24(s0)
 694: 00078713          mv a4,a5
 698: fec42783          lw a5,-20(s0)
 69c: 00f707bb          addw a5,a4,a5
PC→6a0: fef42423        sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```

| | |
|---|---|
| a0 | Input Value |
| a1 | |
| a2 | |
| a3 | |
| a4 | Variable c |
| a5 | sum c+i |
| a6 | |




```
int main(int argc, char *argv[])
{
    int c=0;
    for (int i=0; i<10; i++)
    { c = c + i;}
    printf("Ergebnis ist %i\n",c);
}
```

```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
 674: 03010413          add s0,sp,48
 678: 00050793          mv a5,a0
 67c: fcb43823          sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
 684: fe042423          sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
 68c: 0240006f          j 6b0 <main+0x48>
 690: fe842783          lw a5,-24(s0)
 694: 00078713          mv a4,a5
 698: fec42783          lw a5,-20(s0)
 69c: 00f707bb          addw a5,a4,a5
 6a0: fef42423          sw a5,-24(s0)
PC→6a4: fec42783        lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```

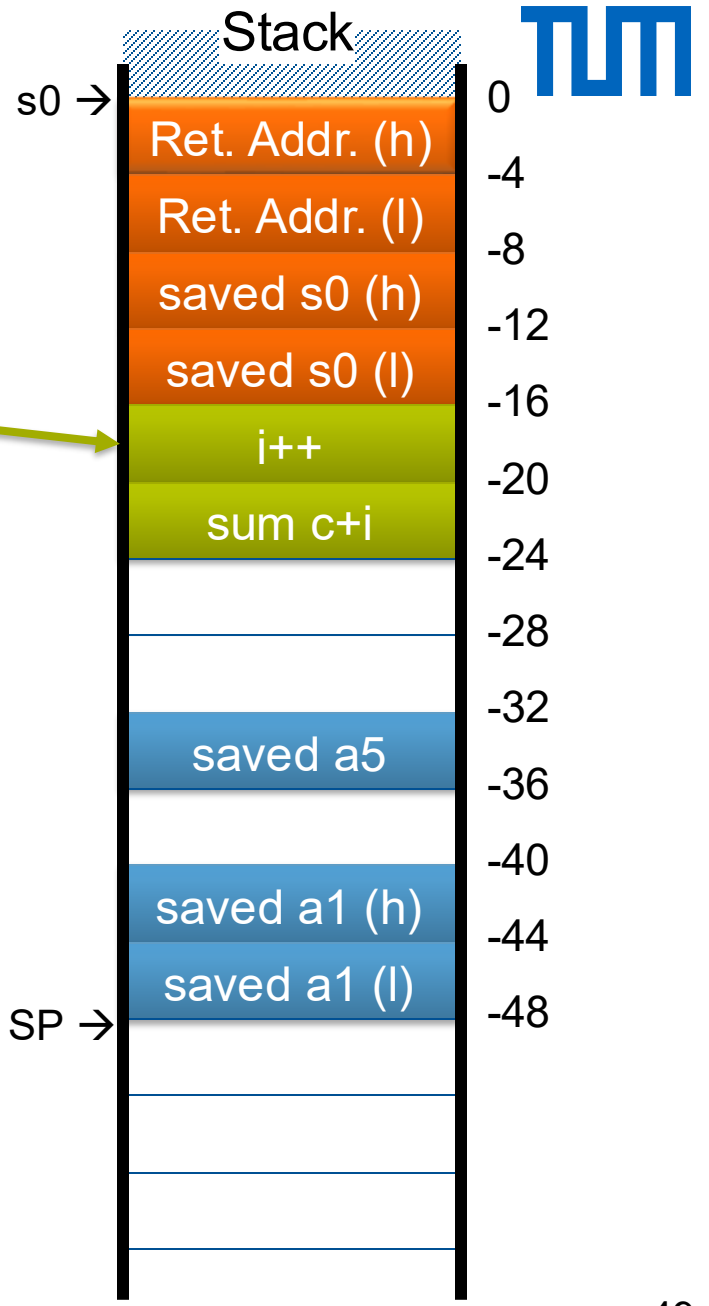



```
int main(int argc, char *argv[])
{
    int c=0;
    for (int i=0; i<10; i++)
    { c = c + i;}
    printf("Ergebnis ist %i\n",c);
}
```

```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
 674: 03010413          add s0,sp,48
 678: 00050793          mv a5,a0
 67c: fcb43823          sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
 684: fe042423          sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
 68c: 0240006f          j 6b0 <main+0x48>
 690: fe842783          lw a5,-24(s0)
 694: 00078713          mv a4,a5
 698: fec42783          lw a5,-20(s0)
 69c: 00f707bb          addw a5,a4,a5
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
PC→6b0: fec42783        lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```

| | |
|---|---|
| a0 | Input Value |
| a1 | |
| a2 | |
| a3 | |
| a4 | Variable c |
| a5 | i++ |
| a6 | |

Check Loop Conditions

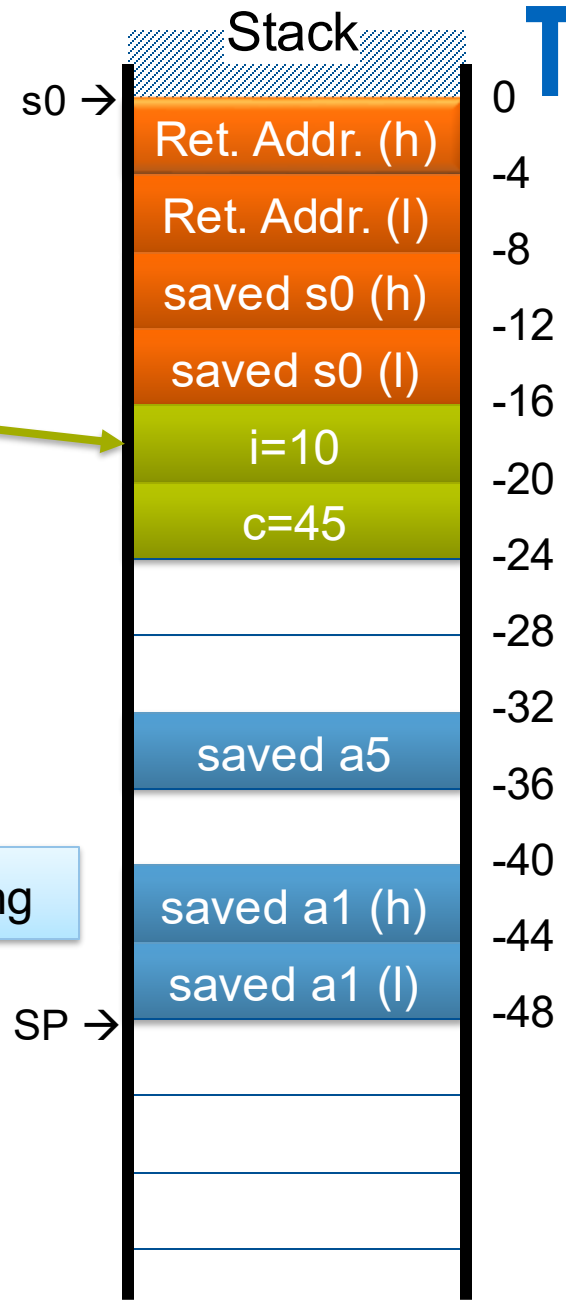


```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
```

```c
int main(int argc, char *argv[])
{
    int c=0;
    for (int i=0; i<10; i++)
    { c = c + i;}
    printf("Ergebnis ist %i\n",c);
}
```

```
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
PC→6c0: fe842783        lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```

| | |
|---|---|
| a0 | Input Value |
| a1 | |
| a2 | |
| a3 | |
| a4 | c=45 (Erg.) |
| a5 | i=10 |
| a6 | |



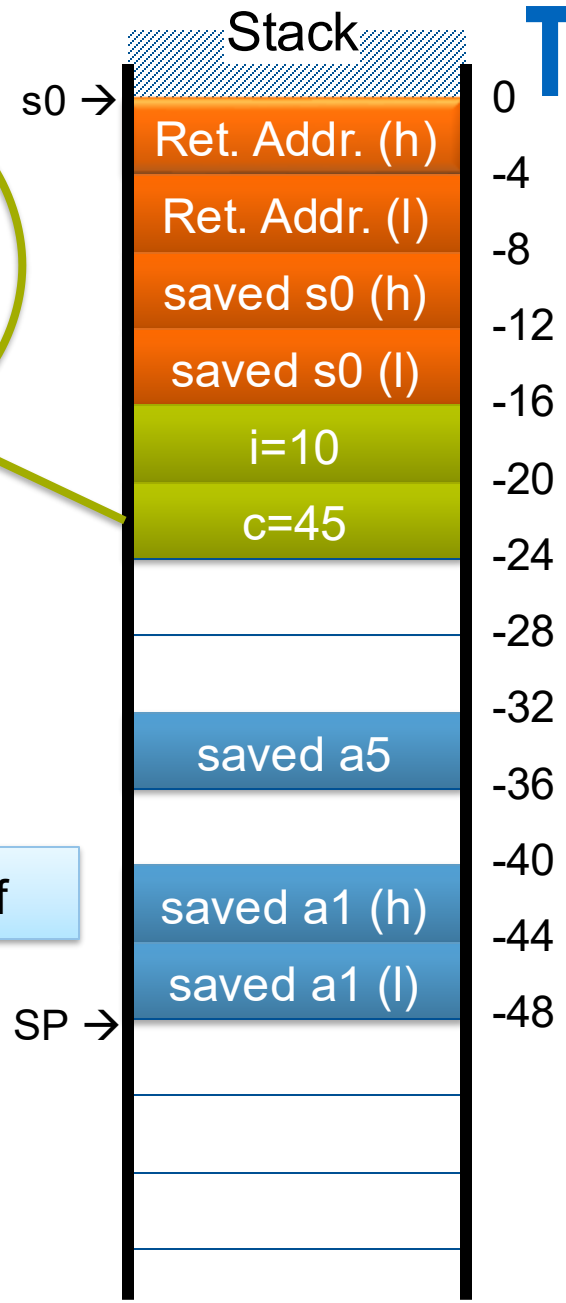

Ende der Berechnung

```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
```

```c
int main(int argc, char *argv[])
{
    int c=0;
    for (int i=0; i<10; i++)
    { c = c + i;}
    printf("Ergebnis ist %i\n",c);
}
```

```
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
PC→6c0: fe842783        lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```

| | |
|---|---|
| a0 | Input Value |
| a1 | c=45 |
| a2 | |
| a3 | |
| a4 | c=45 (Erg.) |
| a5 | c=45 |
| a6 | |




Parameter für printf

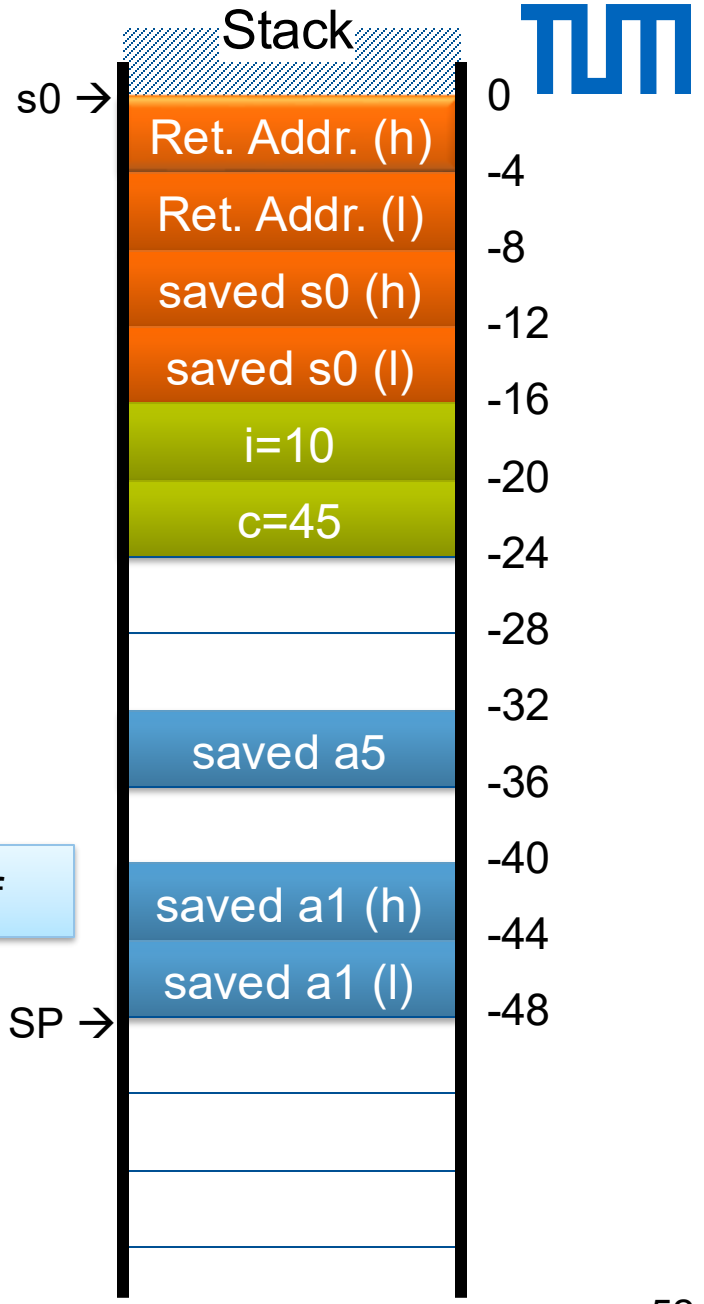
```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
```

```c
int main(int argc, char *argv[])
{
    int c=0;
    for (int i=0; i<10; i++)
    { c = c + i;}
    printf("Ergebnis ist %i\n",c);
}
```

```
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
PC→6c8: 00000517        auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```

| | |
|---|---|
| a0 | 6c8+30=6f8 |
| a1 | c=45 |
| a2 | |
| a3 | |
| a4 | c=45 (Erg.) |
| a5 | c=45 |
| a6 | |

Parameter für printf



# Textkonstanten

Gesuchte Konstant liegt auf 0x6f8 → Eigenes Segment

```
>> objdump -h count
count:     file format elf64-littleriscv
Sections:
…
11 .text         0000013c  00000000000005b0  00000000000005b0  000005b0  2**2
                 CONTENTS, ALLOC, LOAD, READONLY, CODE
12 .rodata       00000019  00000000000006f0  00000000000006f0  000006f0  2**3
                 CONTENTS, ALLOC, LOAD, READONLY, DATA
…
```

Inhalt der Binärdatei ab 0x6f8

```
>> hexdump count -s 0x6f0 -C -n64
000006f0  01 00 02 00 00 00 00 00  45 72 67 65 62 6e 69 73  |........Ergebnis|
00000700  20 69 73 74 20 25 69 0a  00 00 00 00 01 1b 03 3b  | ist %i........;|
00000710  10 00 00 00 01 00 00 00  a4 fe ff ff 28 00 00 00  |............(...|
00000720  10 00 00 00 00 00 00 00  03 7a 52 00 01 7c 01 01  |.........zR..|..|
```

# Textzeichen müssen kodiert werden: ASCII Tabelle

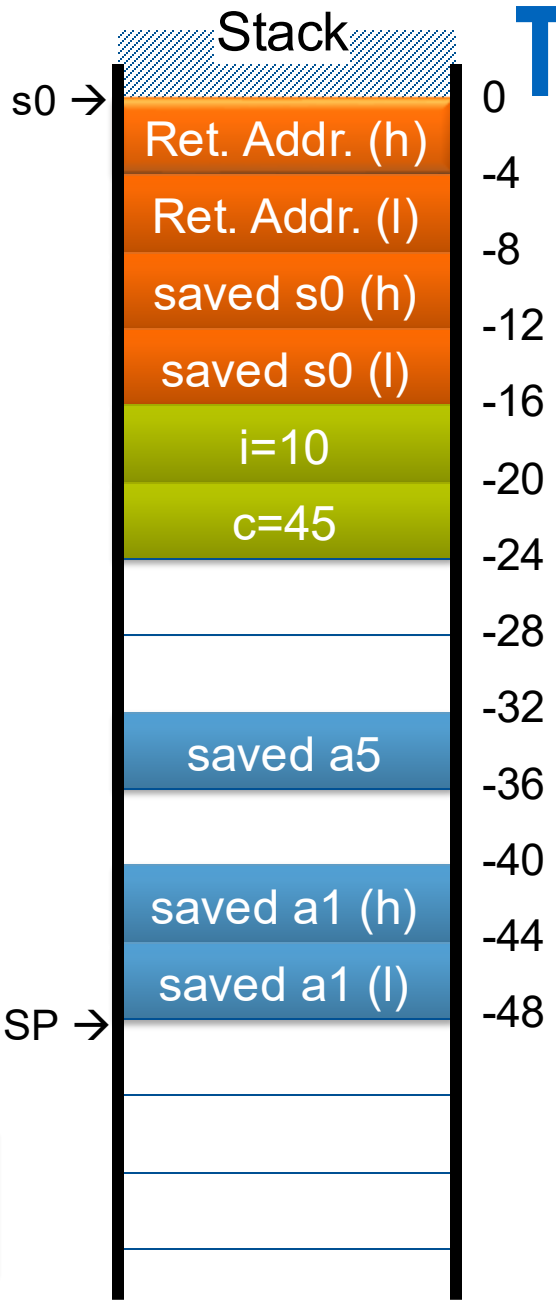
| Decimal | Hexadecimal | Binary | Octal | Char |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | [NULL] |
| 1 | 1 | 1 | 1 | [START OF HEADING] |
| 2 | 2 | 10 | 2 | [START OF TEXT] |
| 3 | 3 | 11 | 3 | [END OF TEXT] |
| 4 | 4 | 100 | 4 | [END OF TRANSMISSION] |
| 5 | 5 | 101 | 5 | [ENQUIRY] |
| 6 | 6 | 110 | 6 | [ACKNOWLEDGE] |
| 7 | 7 | 111 | 7 | [BELL] |
| 8 | 8 | 1000 | 10 | [BACKSPACE] |
| 9 | 9 | 1001 | 11 | [HORIZONTAL TAB] |
| 10 | A | 1010 | 12 | [LINE FEED] |
| 11 | B | 1011 | 13 | [VERTICAL TAB] |
| 12 | C | 1100 | 14 | [FORM FEED] |
| 13 | D | 1101 | 15 | [CARRIAGE RETURN] |
| 14 | E | 1110 | 16 | [SHIFT OUT] |
| 15 | F | 1111 | 17 | [SHIFT IN] |
| 16 | 10 | 10000 | 20 | [DATA LINK ESCAPE] |
| 17 | 11 | 10001 | 21 | [DEVICE CONTROL 1] |
| 18 | 12 | 10010 | 22 | [DEVICE CONTROL 2] |
| 19 | 13 | 10011 | 23 | [DEVICE CONTROL 3] |
| 20 | 14 | 10100 | 24 | [DEVICE CONTROL 4] |
| 21 | 15 | 10101 | 25 | [NEGATIVE ACKNOWLEDGE] |
| 22 | 16 | 10110 | 26 | [SYNCHRONOUS IDLE] |
| 23 | 17 | 10111 | 27 | [ENG OF TRANS. BLOCK] |
| 24 | 18 | 11000 | 30 | [CANCEL] |
| 25 | 19 | 11001 | 31 | [END OF MEDIUM] |
| 26 | 1A | 11010 | 32 | [SUBSTITUTE] |
| 27 | 1B | 11011 | 33 | [ESCAPE] |
| 28 | 1C | 11100 | 34 | [FILE SEPARATOR] |
| 29 | 1D | 11101 | 35 | [GROUP SEPARATOR] |
| 30 | 1E | 11110 | 36 | [RECORD SEPARATOR] |
| 31 | 1F | 11111 | 37 | [UNIT SEPARATOR] |
| 32 | 20 | 100000 | 40 | [SPACE] |
| 33 | 21 | 100001 | 41 | ! |
| 34 | 22 | 100010 | 42 | " |
| 35 | 23 | 100011 | 43 | # |
| 36 | 24 | 100100 | 44 | $ |
| 37 | 25 | 100101 | 45 | % |
| 38 | 26 | 100110 | 46 | & |
| 39 | 27 | 100111 | 47 | ' |
| 40 | 28 | 101000 | 50 | ( |
| 41 | 29 | 101001 | 51 | ) |
| 42 | 2A | 101010 | 52 | * |
| 43 | 2B | 101011 | 53 | + |
| 44 | 2C | 101100 | 54 | , |
| 45 | 2D | 101101 | 55 | - |
| 46 | 2E | 101110 | 56 | . |
| 47 | 2F | 101111 | 57 | / |
| 48 | 30 | 110000 | 60 | 0 |
| 49 | 31 | 110001 | 61 | 1 |
| 50 | 32 | 110010 | 62 | 2 |
| 51 | 33 | 110011 | 63 | 3 |
| 52 | 34 | 110100 | 64 | 4 |
| 53 | 35 | 110101 | 65 | 5 |
| 54 | 36 | 110110 | 66 | 6 |
| 55 | 37 | 110111 | 67 | 7 |
| 56 | 38 | 111000 | 70 | 8 |
| 57 | 39 | 111001 | 71 | 9 |
| 58 | 3A | 111010 | 72 | : |
| 59 | 3B | 111011 | 73 | ; |
| 60 | 3C | 111100 | 74 | < |
| 61 | 3D | 111101 | 75 | = |
| 62 | 3E | 111110 | 76 | > |
| 63 | 3F | 111111 | 77 | ? |
| 64 | 40 | 1000000 | 100 | @ |
| 65 | 41 | 1000001 | 101 | A |
| 66 | 42 | 1000010 | 102 | B |
| 67 | 43 | 1000011 | 103 | C |
| 68 | 44 | 1000100 | 104 | D |
| 69 | 45 | 1000101 | 105 | E |
| 70 | 46 | 1000110 | 106 | F |
| 71 | 47 | 1000111 | 107 | G |
| 72 | 48 | 1001000 | 110 | H |
| 73 | 49 | 1001001 | 111 | I |
| 74 | 4A | 1001010 | 112 | J |
| 75 | 4B | 1001011 | 113 | K |
| 76 | 4C | 1001100 | 114 | L |
| 77 | 4D | 1001101 | 115 | M |
| 78 | 4E | 1001110 | 116 | N |
| 79 | 4F | 1001111 | 117 | O |
| 80 | 50 | 1010000 | 120 | P |
| 81 | 51 | 1010001 | 121 | Q |
| 82 | 52 | 1010010 | 122 | R |
| 83 | 53 | 1010011 | 123 | S |
| 84 | 54 | 1010100 | 124 | T |
| 85 | 55 | 1010101 | 125 | U |
| 86 | 56 | 1010110 | 126 | V |
| 87 | 57 | 1010111 | 127 | W |
| 88 | 58 | 1011000 | 130 | X |
| 89 | 59 | 1011001 | 131 | Y |
| 90 | 5A | 1011010 | 132 | Z |
| 91 | 5B | 1011011 | 133 | [ |
| 92 | 5C | 1011100 | 134 | \ |
| 93 | 5D | 1011101 | 135 | ] |
| 94 | 5E | 1011110 | 136 | ^ |
| 95 | 5F | 1011111 | 137 | _ |
| 96 | 60 | 1100000 | 140 | ` |
| 97 | 61 | 1100001 | 141 | a |
| 98 | 62 | 1100010 | 142 | b |
| 99 | 63 | 1100011 | 143 | c |
| 100 | 64 | 1100100 | 144 | d |
| 101 | 65 | 1100101 | 145 | e |
| 102 | 66 | 1100110 | 146 | f |
| 103 | 67 | 1100111 | 147 | g |
| 104 | 68 | 1101000 | 150 | h |
| 105 | 69 | 1101001 | 151 | i |
| 106 | 6A | 1101010 | 152 | j |
| 107 | 6B | 1101011 | 153 | k |
| 108 | 6C | 1101100 | 154 | l |
| 109 | 6D | 1101101 | 155 | m |
| 110 | 6E | 1101110 | 156 | n |
| 111 | 6F | 1101111 | 157 | o |
| 112 | 70 | 1110000 | 160 | p |
| 113 | 71 | 1110001 | 161 | q |
| 114 | 72 | 1110010 | 162 | r |
| 115 | 73 | 1110011 | 163 | s |
| 116 | 74 | 1110100 | 164 | t |
| 117 | 75 | 1110101 | 165 | u |
| 118 | 76 | 1110110 | 166 | v |
| 119 | 77 | 1110111 | 167 | w |
| 120 | 78 | 1111000 | 170 | x |
| 121 | 79 | 1111001 | 171 | y |
| 122 | 7A | 1111010 | 172 | z |
| 123 | 7B | 1111011 | 173 | { |
| 124 | 7C | 1111100 | 174 | \| |
| 125 | 7D | 1111101 | 175 | } |
| 126 | 7E | 1111110 | 176 | ~ |
| 127 | 7F | 1111111 | 177 | [DEL] |

```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
```

```c
int main(int argc, char *argv[])
{
    int c=0;
    for (int i=0; i<10; i++)
    { c = c + i;}
    printf("Ergebnis ist %i\n",c);
}
```

```
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
PC→6d0: ed1ff0ef        jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```

| | |
|---|---|
| a0 | 6c8+30=6f8 |
| a1 | c=45 |
| a2 | |
| a3 | |
| a4 | c=45 (Erg.) |
| a5 | c=45 |
| a6 | |




Aufruf printf
Symbol im Laufzeit System

# Aufrufkonvention

Soviel wie möglich via Register

- Eingabedaten
- Rückgabewert

Gezielte Aufgaben

- a0-a7: Eingaben
- a0-a1: Ergebnisse
- ra: Rücksprungadresse

Caller-saved

- Aufrufer/Caller muss Werte selbst sichern, dürfen von Callee verändert werden

Callee-saved

- Aufgerufene Funktion darf Werte nicht verändern oder muss sie wieder herstellen

| Register | ABI Name | Description | Saver |
|---|---|---|---|
| x0 | zero | Hard-wired zero | — |
| x1 | ra | Return address | Caller |
| x2 | sp | Stack pointer | Callee |
| x3 | gp | Global pointer | — |
| x4 | tp | Thread pointer | — |
| x5–7 | t0–2 | Temporaries | Caller |
| x8 | s0/fp | Saved register/frame pointer | Callee |
| x9 | s1 | Saved register | Callee |
| x10–11 | a0–1 | Function arguments/return values | Caller |
| x12–17 | a2–7 | Function arguments | Caller |
| x18–27 | s2–11 | Saved registers | Callee |
| x28–31 | t3–6 | Temporaries | Caller |
| f0–7 | ft0–7 | FP temporaries | Caller |
| f8–9 | fs0–1 | FP saved registers | Callee |
| f10–11 | fa0–1 | FP arguments/return values | Caller |
| f12–17 | fa2–7 | FP arguments | Caller |
| f18–27 | fs2–11 | FP saved registers | Callee |
| f28–31 | ft8–11 | FP temporaries | Caller |

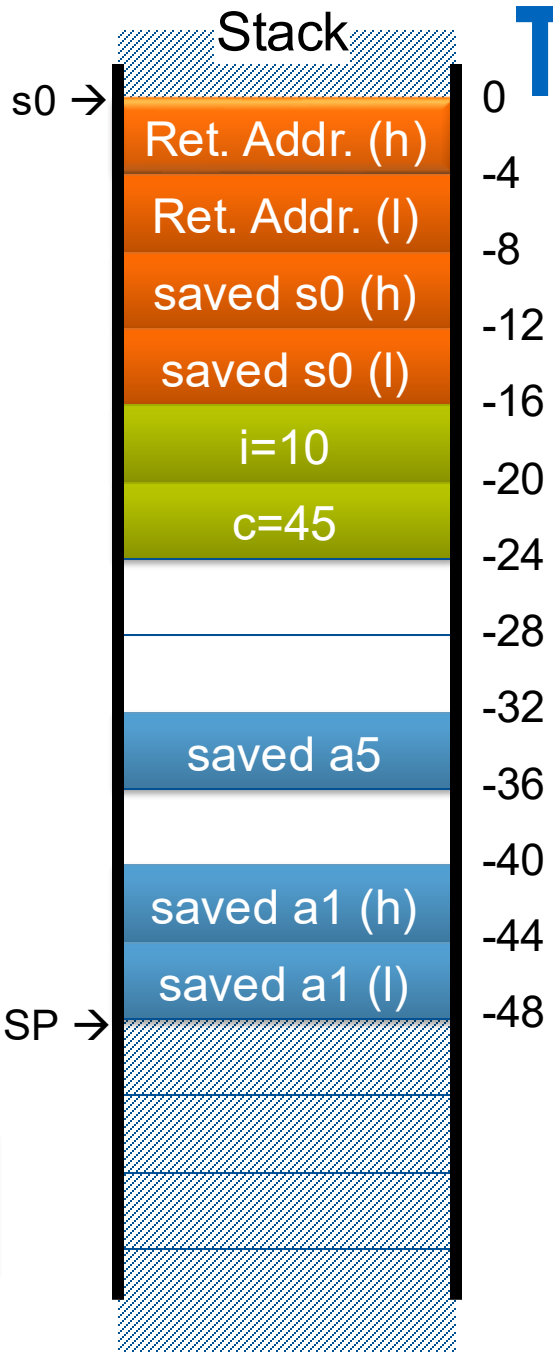
From: https://riscv.org/technical/specifications/

```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
```

```c
int main(int argc, char *argv[])
{
    int c=0;
    for (int i=0; i<10; i++)
    { c = c + i;}
    printf("Ergebnis ist %i\n",c);
}
```

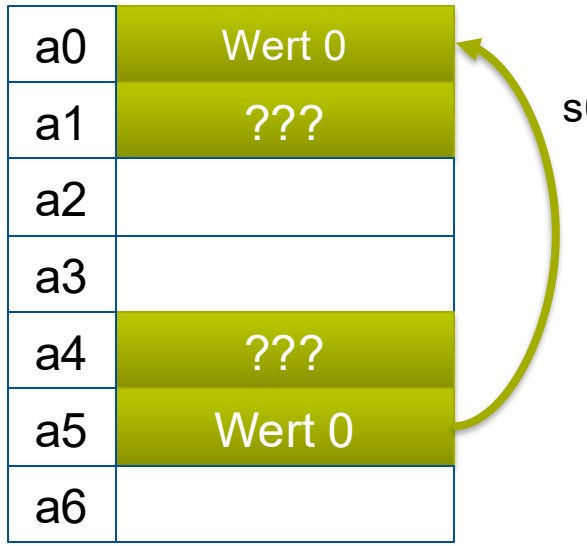
```
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
PC→6d0: ed1ff0ef        jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```

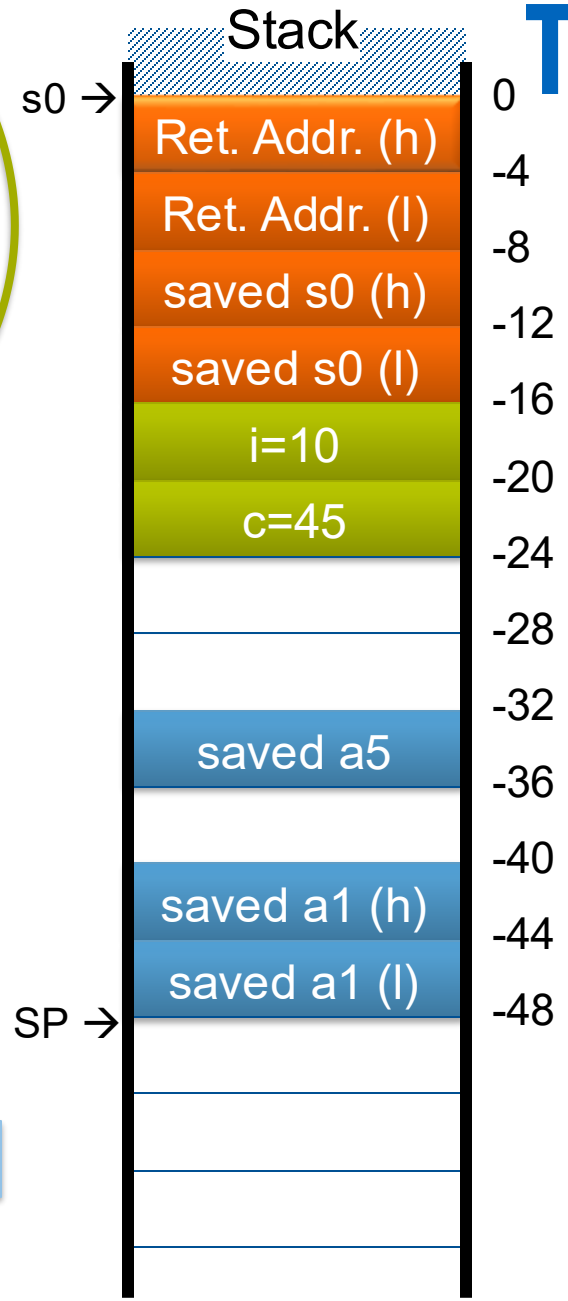
| | |
|---|---|
| a0 | Ergebnis |
| a1 | ??? |
| a2 | |
| a3 | |
| a4 | ??? |
| a5 | ??? |
| a6 | |




Aufruf printf
Symbol in Laufzeit System

```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
 674: 03010413          add s0,sp,48
 678: 00050793          mv a5,a0
 67c: fcb43823          sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
 684: fe042423          sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
 68c: 0240006f          j 6b0 <main+0x48>
 690: fe842783          lw a5,-24(s0)
 694: 00078713          mv a4,a5
 698: fec42783          lw a5,-20(s0)
 69c: 00f707bb          addw a5,a4,a5
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
PC→6d4: 00000793        li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```

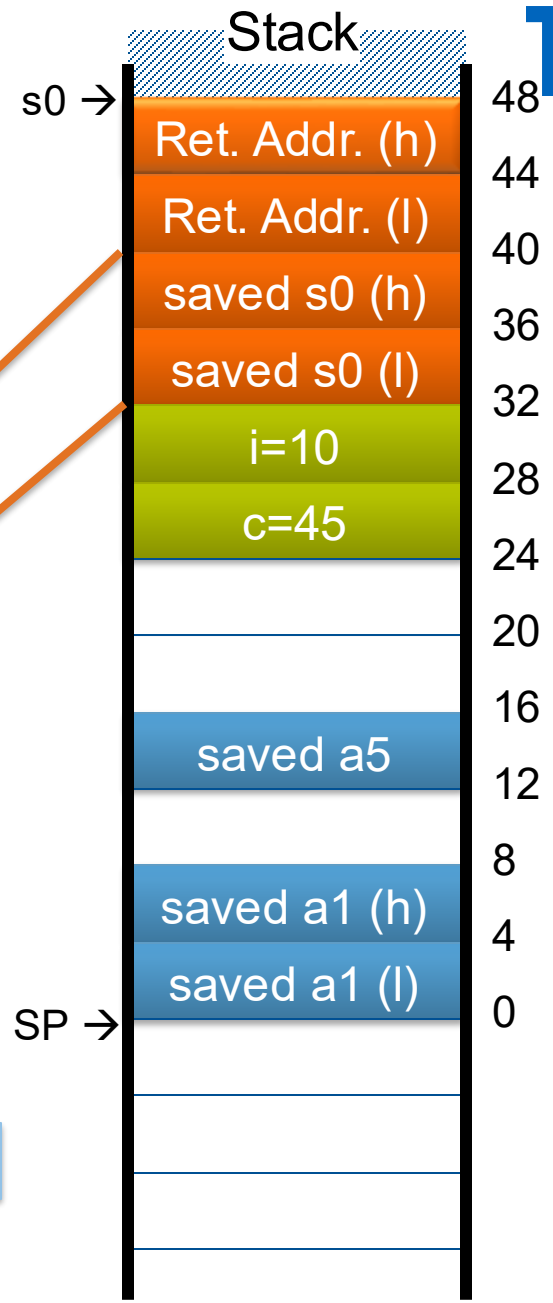



Ergebnis von main = 0

```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
 674: 03010413          add s0,sp,48
 678: 00050793          mv a5,a0
 67c: fcb43823          sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
 684: fe042423          sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
 68c: 0240006f          j 6b0 <main+0x48>
 690: fe842783          lw a5,-24(s0)
 694: 00078713          mv a4,a5
 698: fec42783          lw a5,-20(s0)
 69c: 00f707bb          addw a5,a4,a5
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
PC→6dc: 02813083        ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```

| | |
|---|---|
| a0 | Wert 0 |
| a1 | ??? |
| a2 | |
| a3 | |
| a4 | ??? |
| a5 | Wert 0 |
| a6 | |

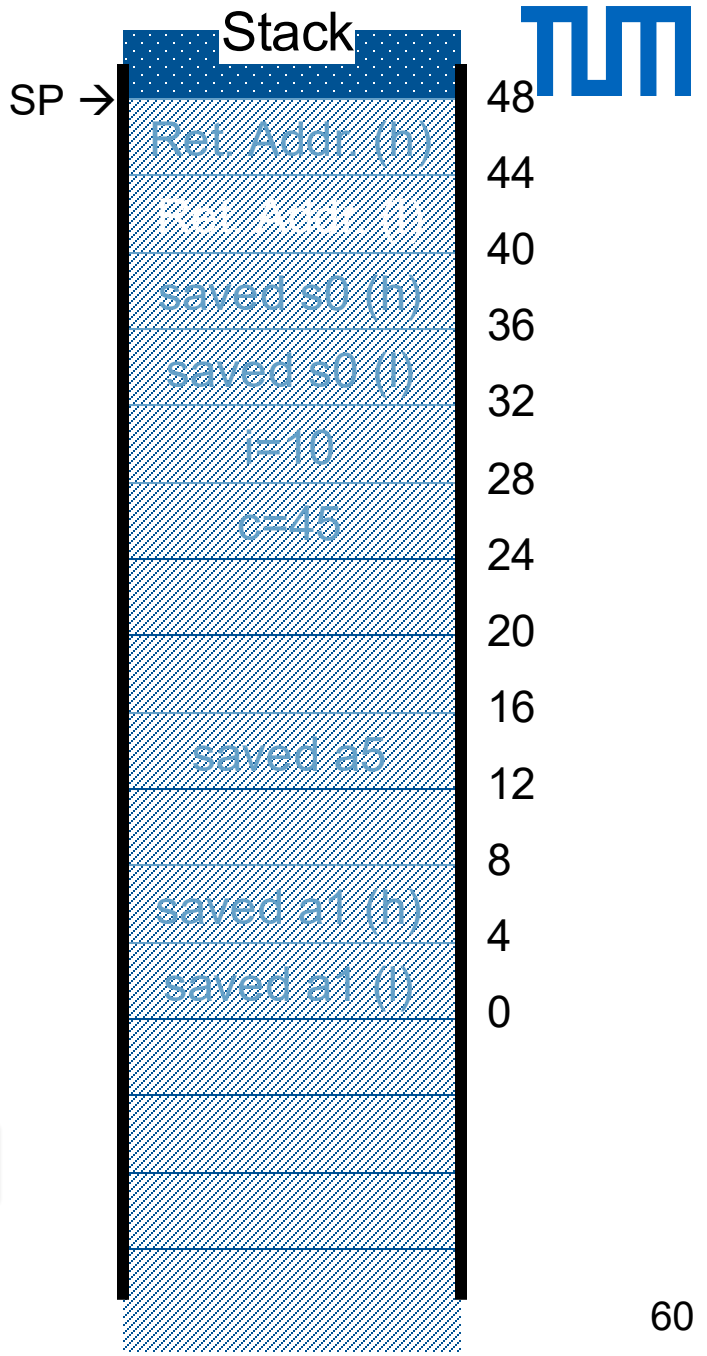



Alte Werte herstellen

```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
 674: 03010413          add s0,sp,48
 678: 00050793          mv a5,a0
 67c: fcb43823          sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
 684: fe042423          sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
 68c: 0240006f          j 6b0 <main+0x48>
 690: fe842783          lw a5,-24(s0)
 694: 00078713          mv a4,a5
 698: fec42783          lw a5,-20(s0)
 69c: 00f707bb          addw a5,a4,a5
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
PC→6e4: 03010113        add sp,sp,48
 6e8: 00008067          ret
```

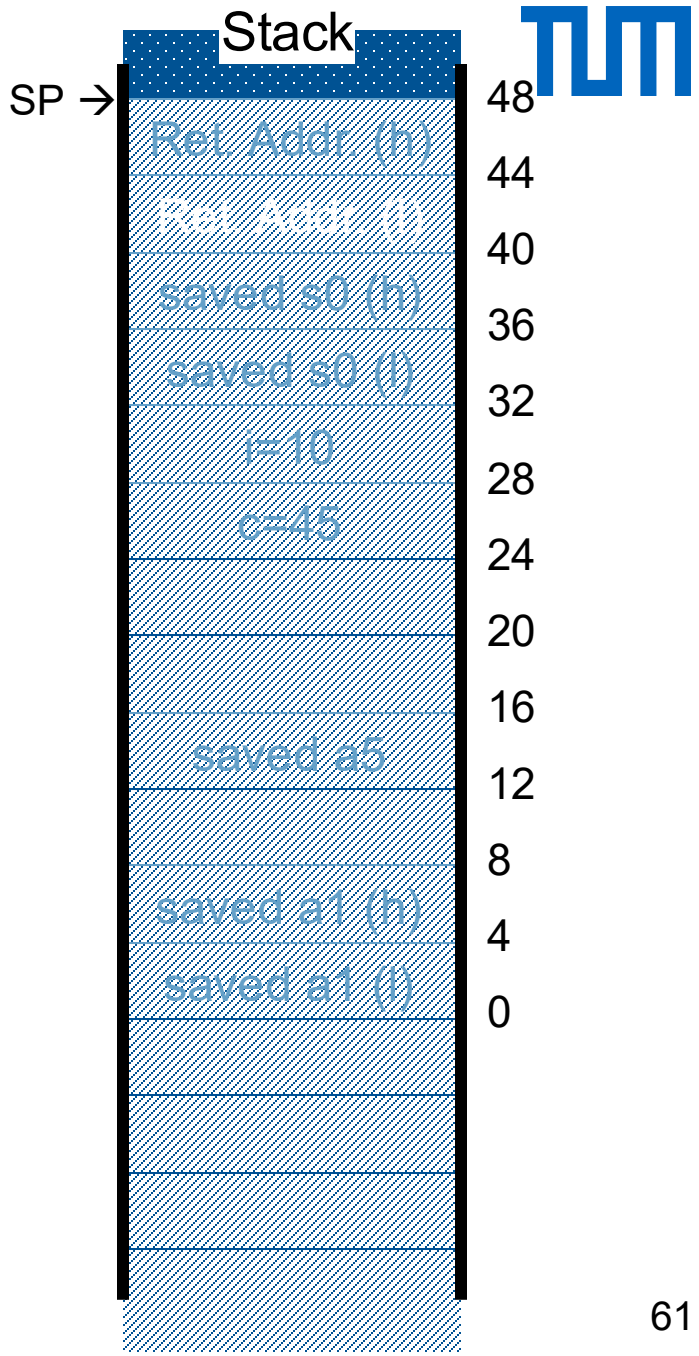
| | |
|---|---|
| a0 | Wert 0 |
| a1 | ??? |
| a2 | |
| a3 | |
| a4 | ??? |
| a5 | Wert 0 |
| a6 | |

Stack “löschen” / freigeben



```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
 674: 03010413          add s0,sp,48
 678: 00050793          mv a5,a0
 67c: fcb43823          sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
 684: fe042423          sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
 68c: 0240006f          j 6b0 <main+0x48>
 690: fe842783          lw a5,-24(s0)
 694: 00078713          mv a4,a5
 698: fec42783          lw a5,-20(s0)
 69c: 00f707bb          addw a5,a4,a5
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
PC→6e8: 00008067        ret
```

| | |
|---|---|
| a0 | Wert 0 |
| a1 | ??? |
| a2 | |
| a3 | |
| a4 | ??? |
| a5 | Wert 0 |
| a6 | |




ret = jalr x0, x1, 0
jalr zero, ra, 0

```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
 674: 03010413          add s0,sp,48
 678: 00050793          mv a5,a0
 67c: fcb43823          sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
 684: fe042423          sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
 68c: 0240006f          j 6b0 <main+0x48>
 690: fe842783          lw a5,-24(s0)
 694: 00078713          mv a4,a5
 698: fec42783          lw a5,-20(s0)
 69c: 00f707bb          addw a5,a4,a5
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```

Prologue
Setup für die Funktion

Variablen zuweisen

Berechnung in der Schleife

Schleife
Bedingung und Sprung

Aufruf ”printf”

Epilogue
Rücksprung

# Beobachtungen

Prologue und Epilogue sind wichtig

- Sicherung von Variablen
- Platz auf dem Stack einrichten
- Einhaltung der Aufrufkonvention

Aufgabe beim Entwurf einer Routine (oder Aufgabe des Compilers)

- Genug Platz reservieren
- Zuweisung von Variablen zu Speicher auf dem Stack
- Belegung von Registern

Aufruf anderer Routinen

- Variablen nach Aufrufkonvention übergeben
- Konstanten evtl. in anderen Segmenten in der Binärdatei
- Bei Rücksprung sind nicht alle Werte erhalten (→ Aufrufkonvention)

Naïve Übersetzung führt zu nicht optimierten Code

```
0000000000000668 <main>:
 668: fd010113          add sp,sp,-48
 66c: 02113423          sd ra,40(sp)
 670: 02813023          sd s0,32(sp)
 674: 03010413          add s0,sp,48
 678: 00050793          mv a5,a0
 67c: fcb43823          sd a1,-48(s0)
 680: fcf42e23          sw a5,-36(s0)
 684: fe042423          sw zero,-24(s0)
 688: fe042623          sw zero,-20(s0)
 68c: 0240006f          j 6b0 <main+0x48>
 690: fe842783          lw a5,-24(s0)
 694: 00078713          mv a4,a5
 698: fec42783          lw a5,-20(s0)
 69c: 00f707bb          addw a5,a4,a5
 6a0: fef42423          sw a5,-24(s0)
 6a4: fec42783          lw a5,-20(s0)
 6a8: 0017879b          addw a5,a5,1
 6ac: fef42623          sw a5,-20(s0)
 6b0: fec42783          lw a5,-20(s0)
 6b4: 0007871b          sext.w a4,a5
 6b8: 00900793          li a5,9
 6bc: fce7dae3          bge a5,a4,690 <main+0x28>
 6c0: fe842783          lw a5,-24(s0)
 6c4: 00078593          mv a1,a5
 6c8: 00000517          auipc a0,0x0
 6cc: 03050513          add a0,a0,48 # 6f8 <_IO_stdin_used+0x8>
 6d0: ed1ff0ef          jal 5a0 <printf@plt>
 6d4: 00000793          li a5,0
 6d8: 00078513          mv a0,a5
 6dc: 02813083          ld ra,40(sp)
 6e0: 02013403          ld s0,32(sp)
 6e4: 03010113          add sp,sp,48
 6e8: 00008067          ret
```

Prologue
Setup für die Funktion

Variablen zuweisen

Berechnung in der Schleife

Schleife
Bedingung und Sprung

Aufruf ”printf”

Epilogue
Rücksprung

# Programm-Abstraktion

Eingabe: Code

- Kodiert als Text
- Nicht direkt ausführbar

C Code

```
int main(int argc, char *argv[])
{
    int c=0;
    for (int i=0; i<10; i++)
      { c = c + i;}
    printf("Ergebnis ist %i\n",c);
}
```

**Compiler Optionen**

-o = Name der Ausgabedatei
-O<n> = Optimierungsgrad (0,1,2,…)
-g = Extra Symbole

-march = ISA supported

- Übersetzen in ausführbare Datei (Linux, z.B. Ubuntu)

```
>> gcc -o count -O2 -g count.c -march=rv64imafd
```

- Ausführung des Programmes

```
>> ./count
Ergebnis ist 45
>>
```

# Optimierter Code

```
00000000000005c0 <main>:
 5c0:   ff010113        add sp,sp,-16
 5c4:   02d00613        li a2,45
 5c8:   00000597        auipc a1,0x0
 5cc:   0e858593        add a1,a1,232 # 6b0 <_IO_stdin_used+0x8>
 5d0:   00100513        li a0,1
 5d4:   00113423        sd ra,8(sp)
 5d8:   fd9ff0ef        jal 5b0 <__printf_chk@plt>
 5dc:   00813083        ld ra,8(sp)
 5e0:   00000513        li a0,0
 5e4:   01010113        add sp,sp,16
 5e8:   00008067        ret
```

Prologue
und
Epilogue

Beobachtung

- Wesentlich kürzer
- Weniger Platz auf dem Stack
- Leicht andere Reihenfolge

Aber:

- Wo ist die Schleife?

```
int main(int argc, char *argv[])
{
    int c=0;
    for (int i=0; i<10; i++)
    { c = c + i;}
    printf("Ergebnis ist %i\n",c);
}
```

# Programm-Abstraktion

Eingabe: Code

- Kodiert als Text
- Nicht direkt ausführbar

C Code

```
int main(int argc, char *argv[])
{
    int c=0;
    for (int i=0; i<argc; i++)
      { c = c + i;}
    printf("Ergebnis ist %i\n",c);
}
```

**Compiler Optionen**

-o = Name der Ausgabedatei
-O<n> = Optimierungsgrad (0,1,2,…)
-g = Extra Symbole

-march = ISA supported

- Übersetzen in ausführbare Datei (Linux, z.B. Ubuntu)

```
>> gcc -o count2 -O2 -g count2.c -march=rv64imafd
```

- Ausführung des Programmes

```
>> ./count2 1 2 3 4 5 6 7 8 9 (argc=10, 9 arguments + binary name)
Ergebnis ist 45
>>
```

# Optimierter Code mit Variabler Eingabe

```
00000000000005c0 <main>:
 5c0:   ff010113            add sp,sp,-16
 5c4:   00113423            sd ra,8(sp)
 5c8:   02a05c63            blez a0,600 <main+0x40>
 5cc:   00000793            li a5,0
 5d0:   00000613            li a2,0
 5d4:   00f6063b            addw a2,a2,a5
 5d8:   0017879b            addw a5,a5,1
 5dc:   fef51ce3            bne a0,a5,5d4 <main+0x14>
 5e0:   00000597            auipc a1,0x0
 5e4:   0e858593            add a1,a1,232 # 6c8 <_IO_stdin_used+0x8>
 5e8:   00100513            li a0,1
 5ec:   fc5ff0ef            jal 5b0 <__printf_chk@plt>
 5f0:   00813083            ld ra,8(sp)
 5f4:   00000513            li a0,0
 5f8:   01010113            add sp,sp,16
 5fc:   00008067            ret
 600:   0000061             li a2,0
 604:   fddff06f            j 5e0 <main+0x20>
```

Eingabe Parameter
argc → a0

Prologue
und
Epilogue

Berechnung

Schleife

Weniger Platz
auf dem Stack

Abkürzung

Berechnungen
In Registern

Schleife
In Registern

```
int main(int argc, char *argv[])
{
    int c=0;
    for (int i=0; i<argc; i++)
    { c = c + i;}
    printf("Ergebnis ist %i\n",c);
}
```

# Einführung in die Rechnerarchitektur (ERA) IN0004

## Binär-Kodierungen

**Martin Schulz**
**schulzm@in.tum.de**
Chair for Computer Architecture and Parallel Systems
https://www.ce.cit.tum.de/caps/

**Robert Wille**
**robert.wille@tum.de**
Chair for Design Automation
https://www.cda.cit.tum.de/

# Programm-Abstraktion

Eingabe: Code

- Kodiert als Text
- Nicht direkt ausführbar

C Code

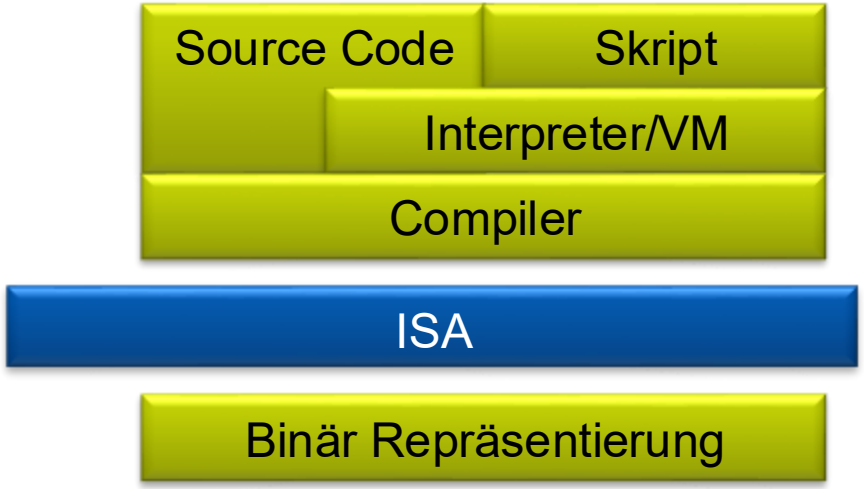
```
int main(int argc, char *argv[])
{
    int c=0;
    for (int i=0; i<argc; i++)
      { c = c + i;}
    printf("Ergebnis ist %i\n",c);
}
```

- Übersetzen in ausführbare Datei (Linux, z.B. Ubuntu)

```
>> gcc -o count2 -O2 -g count2.c
-march=rv64imafd
```

- Ausführung des Programmes

```
>> ./count2 1 2 3 4 5 6 7 8 9
Ergebnis ist 45
>>
```





Binär-Code

# Optimierter Code mit Variabler Eingabe

```
00000000000005c0 <main>:
 5c0:   ff010113            add sp,sp,-16
 5c4:   00113423            sd ra,8(sp)
 5c8:   02a05c63            blez a0,600 <main+0x40>
 5cc:   00000793            li a5,0
 5d0:   00000613            li a2,0
 5d4:   00f6063b            addw a2,a2,a5
 5d8:   0017879b            addw a5,a5,1
 5dc:   fef51ce3            bne a0,a5,5d4 <main+0x14>
 5e0:   00000597            auipc a1,0x0
 5e4:   0e858593            add a1,a1,232 # 6c8 <_IO_stdin_used+0x8>
 5e8:   00100513            li a0,1
 5ec:   fc5ff0ef            jal 5b0 <__printf_chk@plt>
 5f0:   00813083            ld ra,8(sp)
 5f4:   00000513            li a0,0
 5f8:   01010113            add sp,sp,16
 5fc:   00008067            ret
 600:   00000613            li a2,0
 604:   fddff06f            j 5e0 <main+0x20>
```

Berechnung

Schleife

Prologue und Epilogue

Abkürzung

```
int main(int argc, char *argv[])
{
    int c=0;
    for (int i=0; i<argc; i++)
    { c = c + i;}
    printf("Ergebnis ist %i\n",c);
}
```

# Manuelles Disassemblieren

| 31 27 | 26 25 | 24 20 | 19 15 | 14 12 | 11 7 | 6 0 | |
|---|---|---|---|---|---|---|---|
| funct7 | | rs2 | rs1 | funct3 | rd | opcode | R-type |
| imm[11:0] | | | rs1 | funct3 | rd | opcode | I-type |
| imm[11:5] | | rs2 | rs1 | funct3 | imm[4:0] | opcode | S-type |
| imm[12\|10:5] | | rs2 | rs1 | funct3 | imm[4:1\|11] | opcode | B-type |
| imm[31:12] | | | | | rd | opcode | U-type |
| imm[20\|10:1\|11\|19:12] | | | | | rd | opcode | J-type |

5f8: 01010113 add sp,sp,16 → 0000 0001 0000 0001 0000 0001 0001 0011

| | | | | | |
|---|---|---|---|---|---|
| 0000 000 \| 1 0000 | | 00010 | 000 | 00010 | 001 0011 |

| | | | | | |
|---|---|---|---|---|---|
| imm[11:0] | | rs1 | 000 | rd | 0010011 | ADDI |
| 0000000 | rs2 | rs1 | 000 | rd | 0110011 | ADD |

# Aufrufkonvention

Soviel wie möglich via Register

- Eingabedaten
- Rückgabewert

Gezielte Aufgaben

- a0-a7: Eingaben
- a0-a1: Ergebnisse
- ra: Rücksprungadresse

Caller-saved

- Aufrufer/Caller muss Werte selbst sichern, dürfen von Callee verändert werden

Callee-saved

- Aufgerufene Funktion darf Werte nicht verändern oder muss sie wieder herstellen

| Register | ABI Name | Description | Saver |
|---|---|---|---|
| x0 | zero | Hard-wired zero | — |
| x1 | ra | Return address | Caller |
| x2 | sp | Stack pointer | Callee |
| x3 | gp | Global pointer | — |
| x4 | tp | Thread pointer | — |
| x5–7 | t0–2 | Temporaries | Caller |
| x8 | s0/fp | Saved register/frame pointer | Callee |
| x9 | s1 | Saved register | Callee |
| x10–11 | a0–1 | Function arguments/return values | Caller |
| x12–17 | a2–7 | Function arguments | Caller |
| x18–27 | s2–11 | Saved registers | Callee |
| x28–31 | t3–6 | Temporaries | Caller |
| f0–7 | ft0–7 | FP temporaries | Caller |
| f8–9 | fs0–1 | FP saved registers | Callee |
| f10–11 | fa0–1 | FP arguments/return values | Caller |
| f12–17 | fa2–7 | FP arguments | Caller |
| f18–27 | fs2–11 | FP saved registers | Callee |
| f28–31 | ft8–11 | FP temporaries | Caller |

From: https://riscv.org/technical/specifications/

# Manuelles Disassemblieren

| 31 27 | 26 25 | 24 20 | 19 15 | 14 12 | 11 7 | 6 0 | |
|---|---|---|---|---|---|---|---|
| funct7 | | rs2 | rs1 | funct3 | rd | opcode | R-type |
| imm[11:0] | | | rs1 | funct3 | rd | opcode | I-type |
| imm[11:5] | | rs2 | rs1 | funct3 | imm[4:0] | opcode | S-type |
| imm[12\|10:5] | | rs2 | rs1 | funct3 | imm[4:1\|11] | opcode | B-type |
| imm[31:12] | | | | | rd | opcode | U-type |
| imm[20\|10:1\|11\|19:12] | | | | | rd | opcode | J-type |

5f8: 01010113 add sp,sp,16 → 0000 0001 0000 0001 0000 0001 0001 0011

| | | | | | |
|---|---|---|---|---|---|
| 0000 000 \| 1 0000 | | 00010 | 000 | 00010 | 001 0011 |

| | | | | | |
|---|---|---|---|---|---|
| imm[11:0] | | rs1 | 000 | rd | 0010011 | ADDI |
| 0000000 | rs2 | rs1 | 000 | rd | 0110011 | ADD |

Korrekte Disassemblierung: addi sp,sp,16

# Optimierter Code mit Variabler Eingabe

```
00000000000005c0 <main>:
 5c0:   ff010113            add sp,sp,-16
 5c4:   00113423            sd ra,8(sp)
 5c8:   02a05c63            blez a0,600 <main+0x40>
 5cc:   00000793            li a5,0
 5d0:   00000613            li a2,0
 5d4:   00f6063b            addw a2,a2,a5
 5d8:   0017879b            addw a5,a5,1
 5dc:   fef51ce3            bne a0,a5,5d4 <main+0x14>
 5e0:   00000597            auipc a1,0x0
 5e4:   0e858593            add a1,a1,232 # 6c8 <_IO_stdin_used+0x8>
 5e8:   00100513            li a0,1
 5ec:   fc5ff0ef            jal 5b0 <__printf_chk@plt>
 5f0:   00813083            ld ra,8(sp)
 5f4:   00000513            li a0,0
 5f8:   01010113            add sp,sp,16
 5fc:   00008067            ret
 600:   00000613            li a2,0
 604:   fddff06f            j 5e0 <main+0x20>
```

Berechnung

Schleife

Prologue und Epilogue

Abkürzung

Generiert mit:

objdump –d count

```
int main(int argc, char *argv[])
{
    int c=0;
    for (int i=0; i<argc; i++)
    { c = c + i;}
    printf("Ergebnis ist %i\n",c);
}
```

# Optimierter Code mit Variabler Eingabe

```
00000000000005c0 <main>:
 5c0:   ff010113            addi sp,sp,-16
 5c4:   00113423            sd ra,8(sp)
 5c8:   02a05c63            bge zero,a0,600 <main+0x40>
 5cc:   00000793            addi a5,zero,0
 5d0:   00000613            addi a2,zero,0
 5d4:   00f6063b            addw a2,a2,a5
 5d8:   0017879b            addiw a5,a5,1
 5dc:   fef51ce3            bne a0,a5,5d4 <main+0x14>
 5e0:   00000597            auipc a1,0x0
 5e4:   0e858593            addi a1,a1,232 # 6c8 <_IO_stdin_used+0x8>
 5e8:   00100513            addi a0,zero,1
 5ec:   fc5ff0ef            jal ra,5b0 <__printf_chk@plt>
 5f0:   00813083            ld ra,8(sp)
 5f4:   00000513            addi a0,zero,0
 5f8:   01010113            addi sp,sp,16
 5fc:   00008067            jalr zero,0(ra)
 600:   00000613            addi a2,zero,0
 604:   fddff06f            jal zero,5e0 <main+0x20>
```

Berechnung

Schleife

Prologue und Epilogue

Abkürzung

Generiert mit:

objdump –d count -M no-aliases

```
int main(int argc, char *argv[])
{
    int c=0;
    for (int i=0; i<argc; i++)
    { c = c + i;}
    printf("Ergebnis ist %i\n",c);
}
```

# Einführung in die Rechnerarchitektur (ERA) IN0004

## Die von Neumann Architektur

**Martin Schulz**
**schulzm@in.tum.de**
Chair for Computer Architecture and Parallel Systems
https://www.ce.cit.tum.de/caps/

**Robert Wille**
**robert.wille@tum.de**
Chair for Design Automation
https://www.cda.cit.tum.de/

# John von Neumann

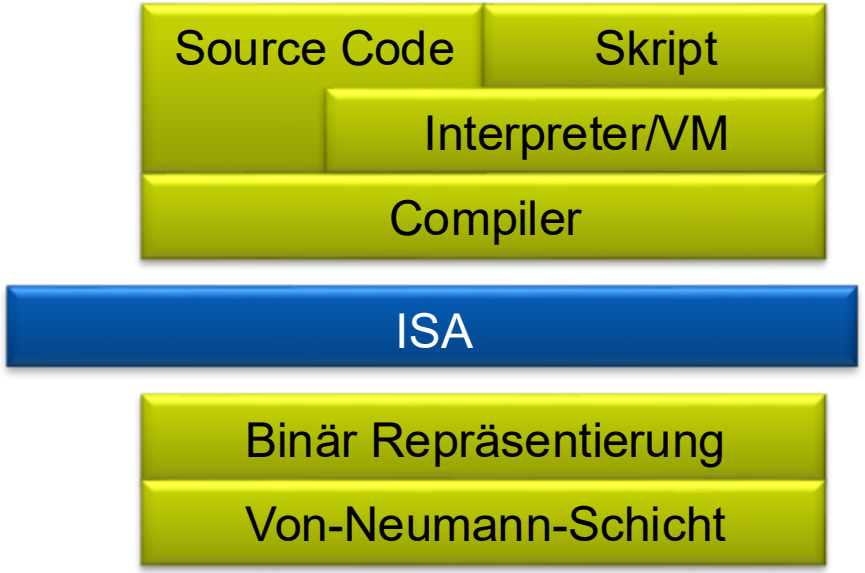
Geboren 1903 in Budapest

- Mathematiker und Physiker
- 1927 Habilitiation in Göttingen
- 1929 Professor in Princeton
- 1933 Ruf auf Lebenszeit

Teil des Manhattan Projekts

- Arbeiten in (u.a.)
  - Quantenlogik
  - Merge Sort
  - Rechnerarchitektur
  - 1952 IAS/Princeton Maschine

Gestorben 1957 in Washington DC



# Von-Neumann Architekturkonzept

Von Neumann Architektur beschrieben
in einem Paper über EDVAC

- EDVAC gebaut in Los Alamos NL (1945)
  Electronic Discrete Variable Automatic Computer
- Nachfolger der ENIAC
- Entwickler der EDVAC:
  John Mauchly & J. Presper Eckert

Papers by Burks, Goldstine, von Neumann:

- First Draft of a Report on the EDVAC, 1945
  https://library.si.edu/digital-library/book/firstdraftofrepo00vonn
- Preliminary discussion of the logical design of
  an electronic computing instrument, 1946
  https://library.ias.edu/files/Prelim_Disc_Logical_Design.pdf

Mehrfach abgedruckt

- Taub: Collected works of J. v. Neumann, 1962
- DATAMATION, Vol 8, Nr. 10, pp. 36-41, 1962
- Bell, Newell: Computer Structures, Readings and examples, Mac Graw Hill, 1971, pp. 92-119

# Von-Neumann Architekturkonzept

1. Die Struktur des Rechners ist unabhängig vom bearbeiteten Problem (Programmsteuerung!)
2. Rechner besteht aus vier Werken
   - Haupt- bzw. Arbeitsspeicher
     (speichert Programme und Daten)
   - Leitwerk
     (arbeitet Befehlszyklus ab)
   - Rechenwerk
     (führt arithmetische Operationen aus)
   - Ein/Ausgabewerk inklusive Sekundärspeicher
     (kommuniziert mit Umgebung, inkl. „Langfrist“-Speicher)

# Von-Neumann Architekturkonzept

1. Die Struktur des Rechners ist unabhängig vom bearbeiteten Problem (Programmsteuerung!)
2. Rechner besteht aus vier Werken
3. Der Hauptspeicher ist in Zellen gleicher Größe geteilt, die durch fortlaufende Nummern (Adressen) bezeichnet werden

Hauptspeicher

| Adresse | Wert |
|---|---|
| 000 | |
| 001 | |
| 010 | |
| 011 | |
| 100 | |
| 101 | |
| 110 | |
| 111 | |

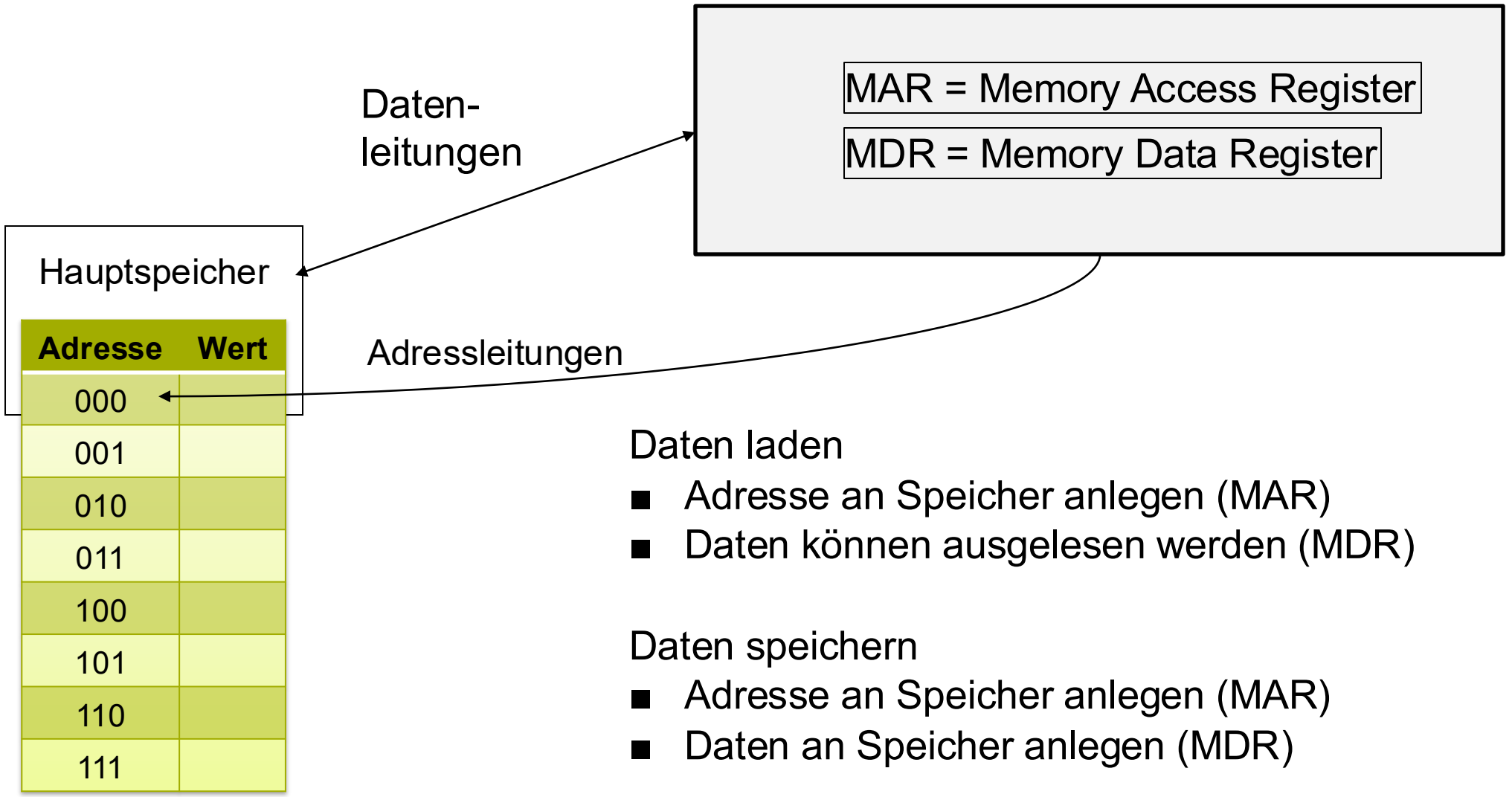
Speicher (siehe Vorlesung letzte Woche)
- Menge von Zellen fester Wortlänge
- Jede Zelle hat eine Adresse

Jede Zelle hat zwei Eigenschaften
- Adresse und Wert

Feste Adresslänge
- Beispiel: 3 bit
- Legt maximale Speichergröße fest

„Random Access Memory“ (RAM)

# Zugriff auf Daten im Speicher




Daten laden

- Adresse an Speicher anlegen (MAR)
- Daten können ausgelesen werden (MDR)

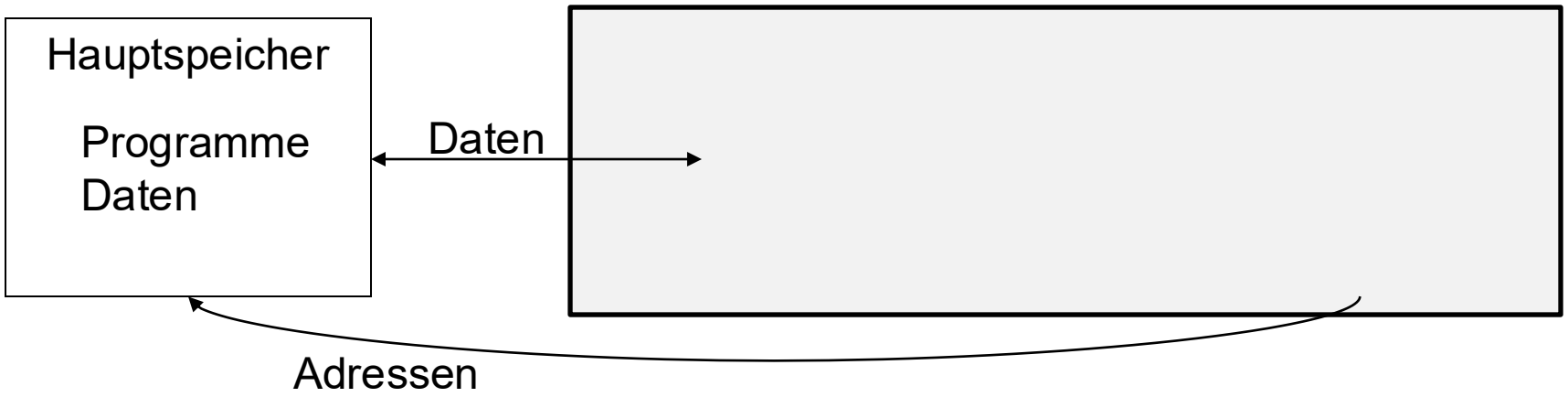
Daten speichern

- Adresse an Speicher anlegen (MAR)
- Daten an Speicher anlegen (MDR)

# Von-Neumann Architekturkonzept

1. Die Struktur des Rechners ist unabhängig vom bearbeiteten Problem (Programmsteuerung!)
2. Rechner besteht aus vier Werken
3. Der Hauptspeicher ist in Zellen gleicher Größe geteilt, die durch fortlaufende Nummern (Adressen) bezeichnet werden
4. Programm und Daten stehen in demselben Speicher (Hauptspeicher) und können durch die Maschine verändert werden

# Von-Neumann Konzept:
# Programme und Daten im Speicher




Kodierung von Daten

- Siehe Binärkodierung in der ersten Vorlesung

Kodierung von Befehlen

- Siehe ISA/Assembler Kodierung

# Von-Neumann Konzept: Design Abwägungen

Programme und Daten liegen im selben Speicher

- Heute oft getrennt verwaltet
- Veränderbare/nicht veränderbare Speicherbereiche
  aka. Princeton Architektur

Alternativen

- Programme und Daten in verschiedenen Speichern
  aka. Harvard Architektur
- Meist Mischform: gemeinsamer Hauptspeicher
  Aber getrennte Zwischenspeicher-Speicher

Konsequenz: die Maschine kann Programme und Daten verändern

- Wichtig für Betriebssysteme (Laden von Programmen)
- Anpassung von Programmen beim Linken
- Aber: (fast) keine selbstmodifizierenden Programme mehr
  - Stattdessen bedingte Sprünge und Adressmodifikation
  - Immer noch zu finden in Viren und dynamischen Werkzeugen

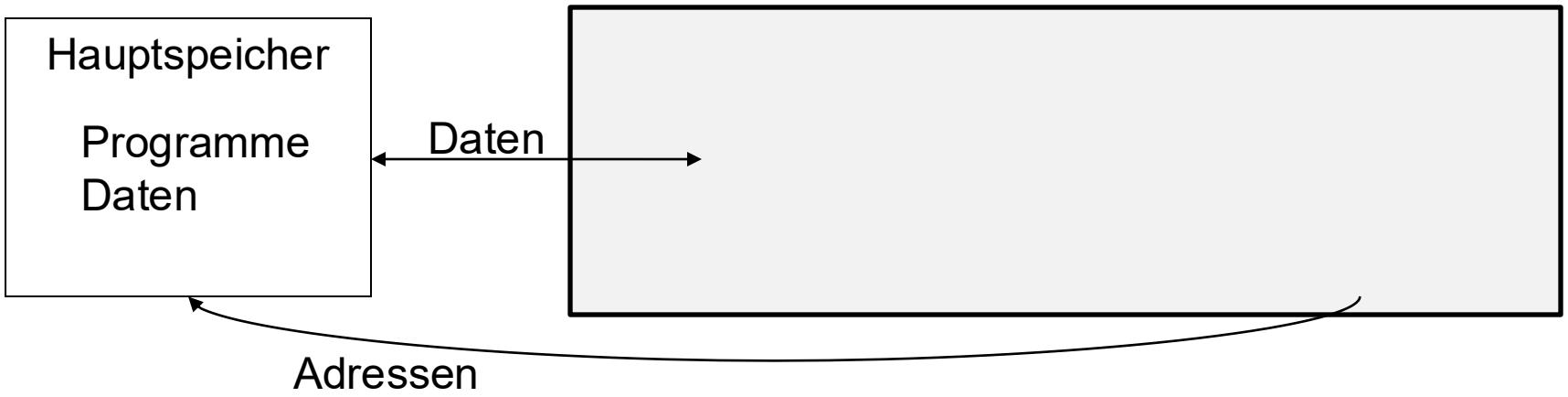
# Von-Neumann Architekturkonzept

1. Die Struktur des Rechners ist unabhängig vom bearbeiteten Problem (Programmsteuerung!)
2. Rechner besteht aus vier Werken
3. Der Hauptspeicher ist in Zellen gleicher Größe geteilt, die durch fortlaufende Nummern (Adressen) bezeichnet werden
4. Programm und Daten stehen in demselben Speicher (Hauptspeicher) und können durch die Maschine verändert werden
5. Die Maschine benutzt Binärcodes; Zahlen werden dual dargestellt

# Von-Neumann Architekturkonzept

1. Die Struktur des Rechners ist unabhängig vom bearbeiteten Problem (Programmsteuerung!)
2. Rechner besteht aus vier Werken
3. Der Hauptspeicher ist in Zellen gleicher Größe geteilt, die durch fortlaufende Nummern (Adressen) bezeichnet werden
4. Programm und Daten stehen in demselben Speicher (Hauptspeicher) und können durch die Maschine verändert werden
5. Die Maschine benutzt Binärcodes; Zahlen werden dual dargestellt
6. Das Programm besteht aus einer Folge von Befehlen (Instruktionen)
   - I.a. nacheinander gespeichert (aufsteigende Adressen)
   - I.a. nacheinander ausgeführt (sequentiell)

# Von-Neumann Konzept:
# Programme und Daten im Speicher




Kodierung von Daten

- Siehe Binärkodierung in der ersten Vorlesung

Kodierung von Befehlen

- Siehe ISA/Assembler Kodierung

Kette von Befehlen → Kodierung von Programmen

# Programm-Darstellung und Bearbeitung

Das Programm besteht aus einer Folge von Befehlen

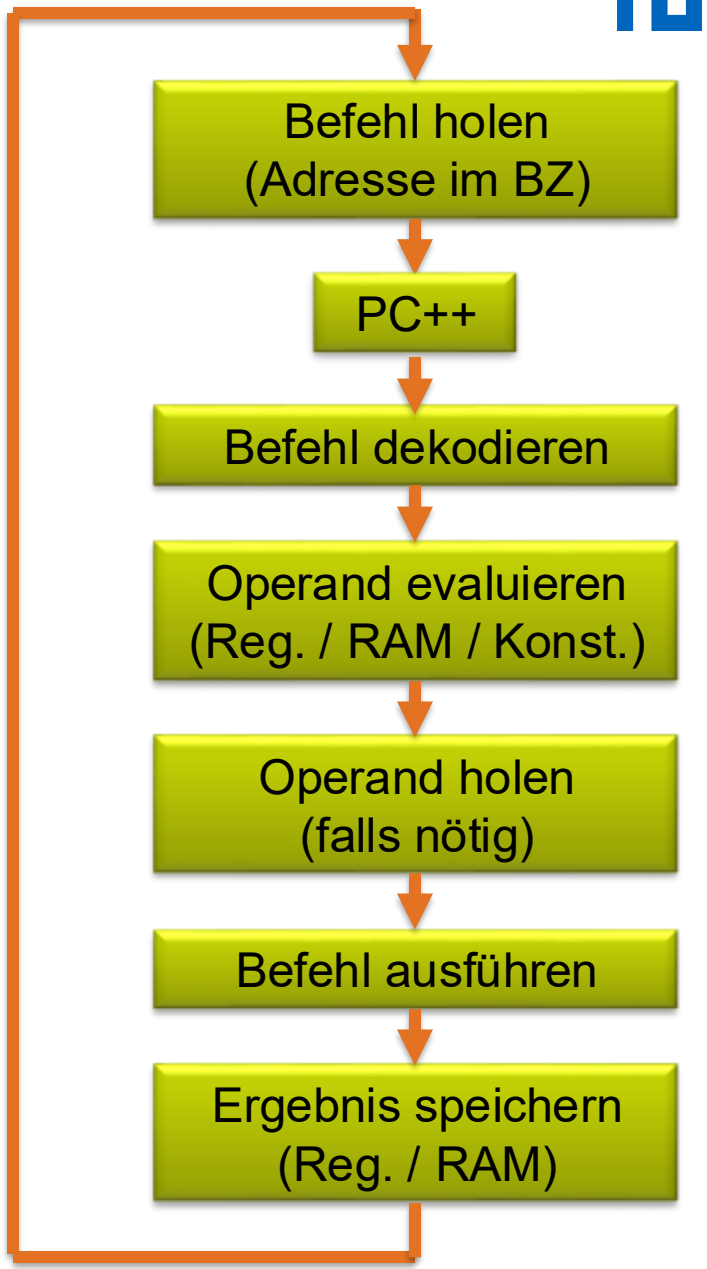
- Befehle beschreiben einen sequentiellen Auftrag
- Befehle sind in Aufschreibungsreihenfolge auszuführen
- Abweichen von dieser Reihenfolge durch Sprünge möglich

Leitwerk führt die Programmausführung durch

- Zu jedem Zeitpunkt führt der Prozessor nur einen Befehl aus
  - Neuere Architekturen können hier anders sein
- Adresse des Befehles wird im Befehlszähler (BZ) gehalten
  - Englisch: Program-Counter (PC)
- Befehl wird aus Speicher ins Instruktionsregister (IR) geladen
- Anschließend Dekodierung / Ausführung

Nach der Ausführung

- Inkrementieren des Befehlszähler (nächster Befehl)
- Setzen auf einen neuen Wert (Sprung)



# Mini Beispiel von Vorlesung 2

Auslesen mit `objdump`

- Alle „Headers“ (Dateiteile) mit „-h“
- Codiertes Programm mit „-d“ = Dissassemblierung
- Codiertes Programm plus Bezug auf Quellcode mit „-S“

```
    c = c + i;
684: fe842783        lw a5,-24(s0)
688: 873e            mv a4,a5
68a: fec42783        lw a5,-20(s0)
68e: 9fb9            addw a5,a5,a4
690: fef42423        sw a5,-24(s0)
```

Operand 1 ermitteln

Operand 2 ermitteln

Addition

Ergebnis abspeichern

Fortlaufende Adressen

Binär-Code = Maschinensprache in ausführbarer Form

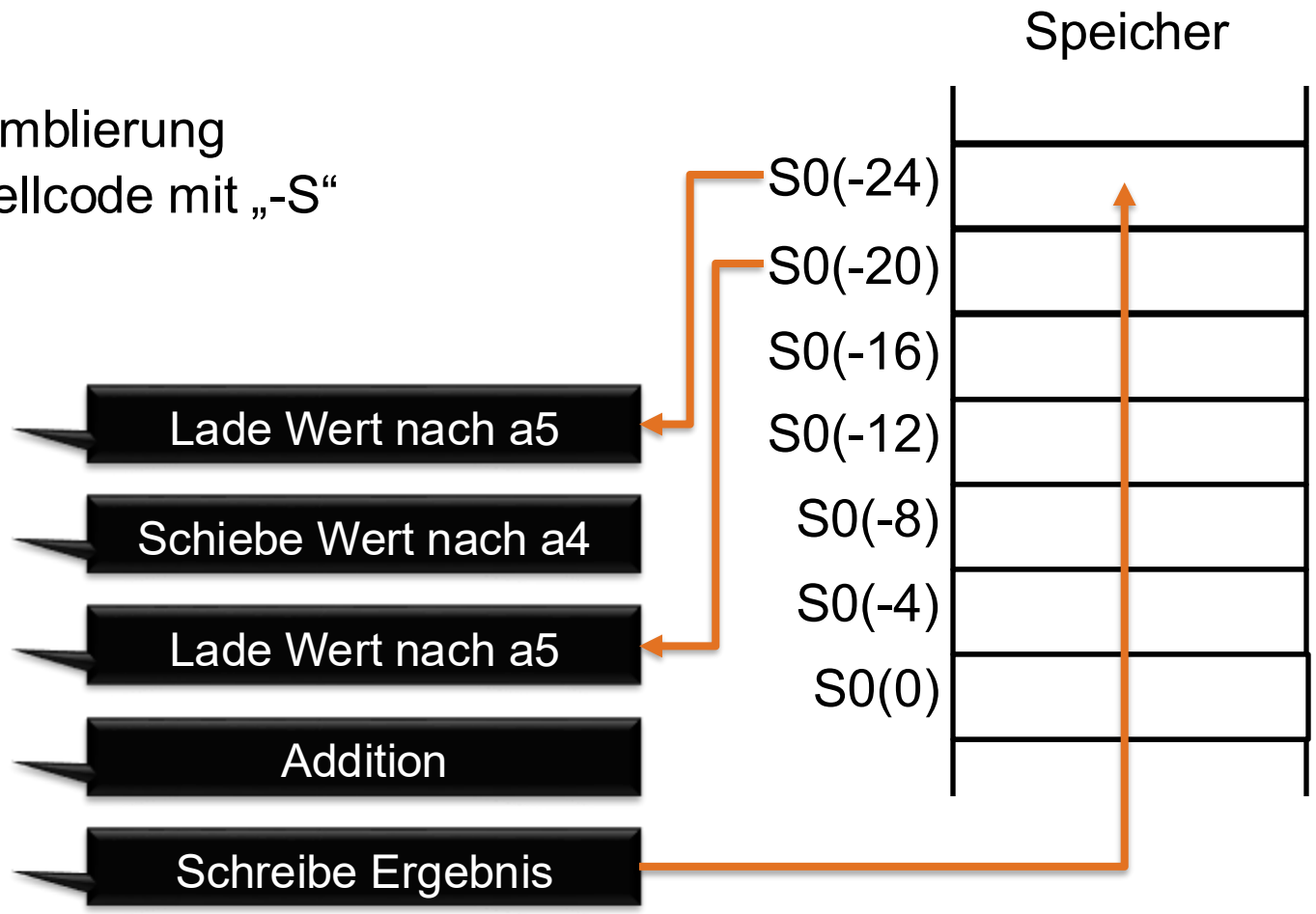
Assembler-Code = Maschinensprache in lesbarer Form

# Mini Beispiel von Vorlesung 2

Auslesen mit `objdump`

- Alle „Headers“ (Dateiteile) mit „-h“
- Codiertes Programm mit „-d“ = Dissassemblierung
- Codiertes Programm plus Bezug auf Quellcode mit „-S“

```
    c = c + i;

684: fe842783        lw a5,-24(s0)
688: 873e            mv a4,a5
68a: fec42783        lw a5,-20(s0)
68e: 9fb9            addw a5,a5,a4
690: fef42423        sw a5,-24(s0)
```



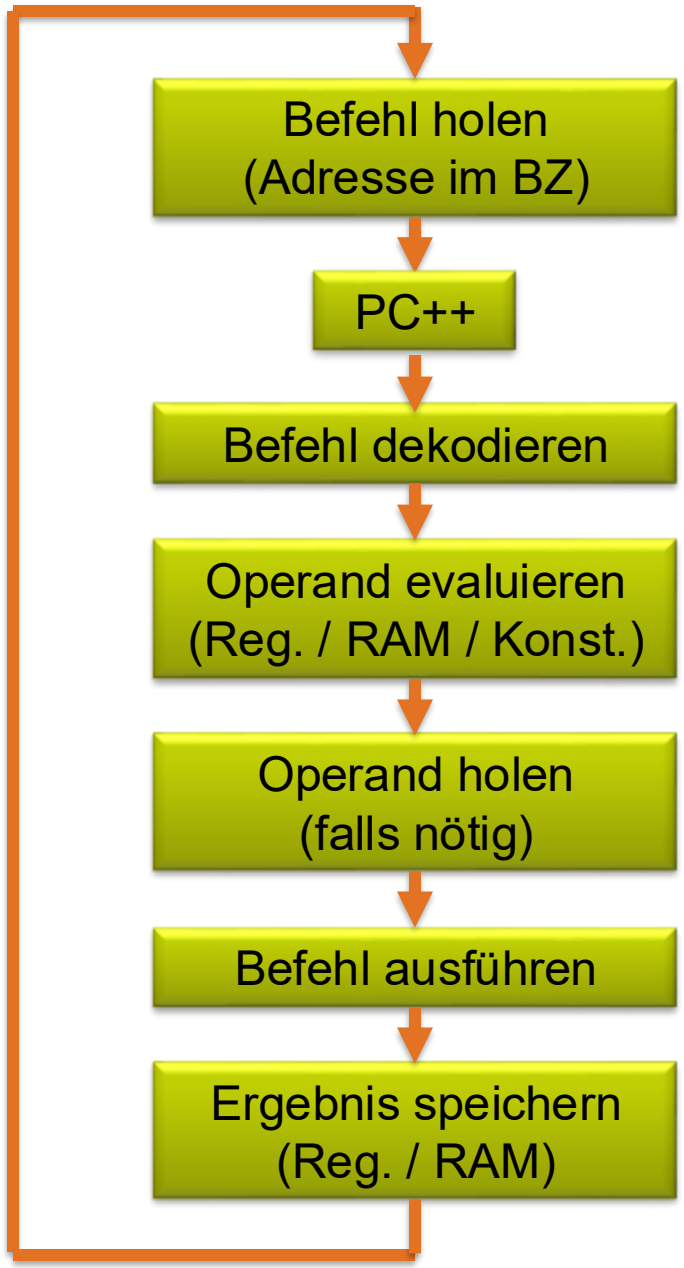

Alle Operationen auf „Word“ Basis, d.h., 32bit („w“ Version der Befehle)

# Instruktionszyklus

Sequentielle Abarbeitung von Befehlen

- Binärform aller Befehle im Speicher
- Befählszähler (BZ) / **Program Counter (PC)**
  - Adresse für den nächsten Befehl

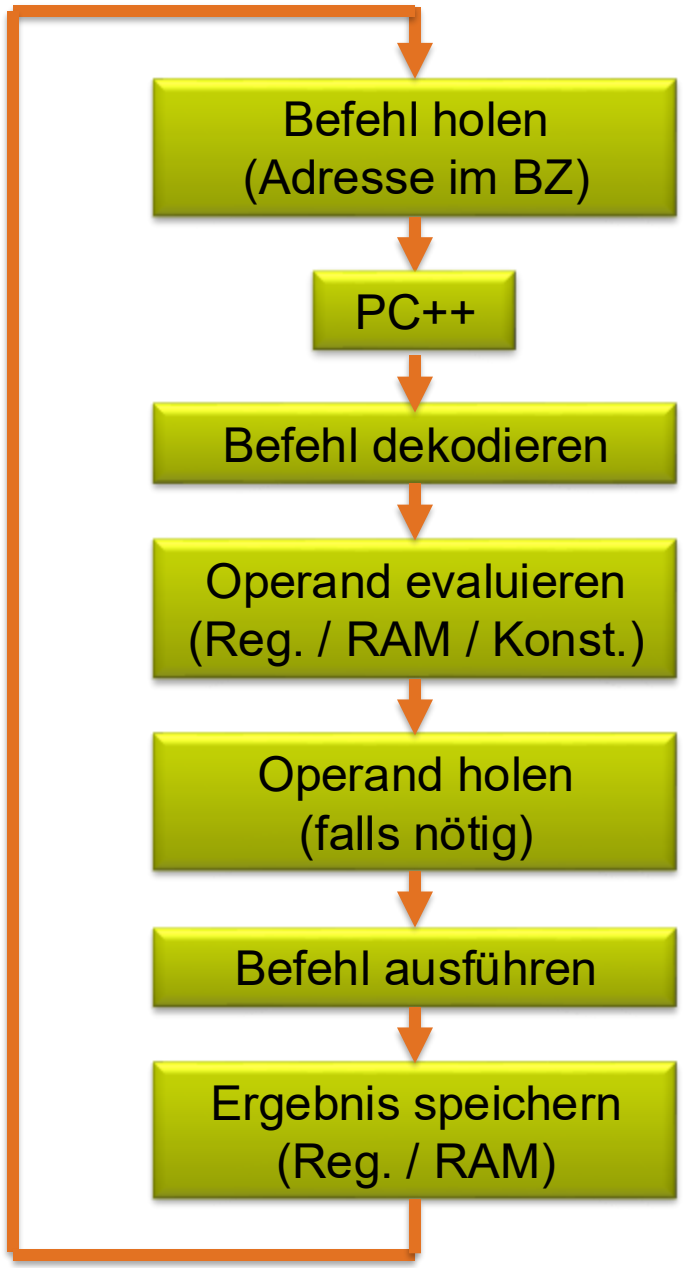
```
    c = c + i;

684: fe842783       lw a5,-24(s0)

688: 873e           mv a4,a5

68a: fec42783       lw a5,-20(s0)

68e: 9fb9           addw a5,a5,a4

690: fef42423       sw a5,-24(s0)
```



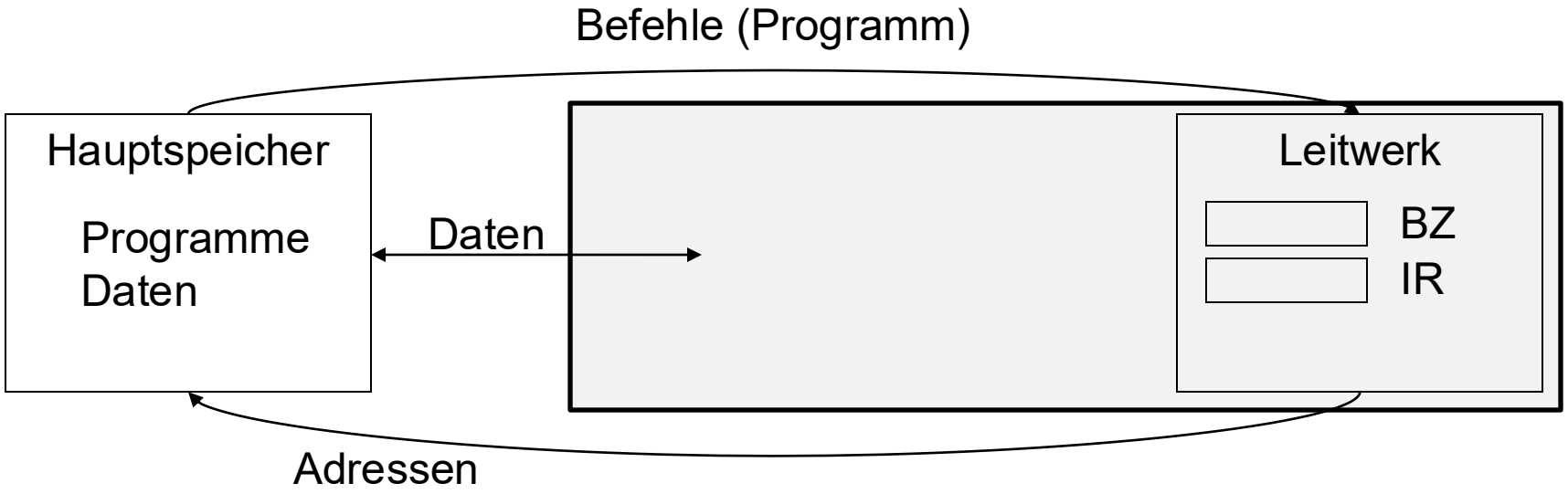
# Instruktionszyklus / Beispiel

```
      c = c + i;
684:  fe842783        lw a5,-24(s0)
688:  873e            mv a4,a5
68a:  fec42783        lw a5,-20(s0)
68e:  9fb9            addw a5,a5,a4
690:  fef42423        sw a5,-24(s0)
```

| PC | geladener Befehl | Speicher Adresse | Speicher Wert | Kommentar |
|---|---|---|---|---|
| 684 | - | PC | - | Anfrage an Speicher für Befehl |
| 688 | - | - | Code(lw) | PC erhöhen → nächster Befehl |
| 688 | lw | - | - | Dekodierung |
| 688 | lw | s0-24 | - | Anfrage Operand 1 |
| 688 | lw | - | [s0-24] | Wert laden und nach a5 schreiben |
| 688 | lw | PC | - | Anfrage an Speicher für Befehl |
| 68a | lw | - | Code(mv) | PC erhöhen → nächster Befehl |
| 68a | mv | - | - | Dekodierung |
| 68a | mv | - | - | Wert von a5 nach a4 schreiben |
| 68a | mv | PC | - | Anfrage an Speicher für Befehl |
| 68e | mv | - | Code(lw) | PC erhöhen → nächster Befehl |
| 68e | lw | - | - | Dekodierung |
| 68e | lw | s0-20 | - | Anfrage Operand 1 |
| 68e | lw | - | [s0-20] | Wert laden und nach a5 schreiben |
| 68e | lw | PC | - | Anfrage an Speicher für Befehl |
| 690 | lw | - | Code(addw) | PC erhöhen → nächster Befehl |
| 690 | addw | - | - | Dekodierung |
| 690 | addw | - | - | Addition, Ergebnis nach a5 schreiben |
| … | | | | |



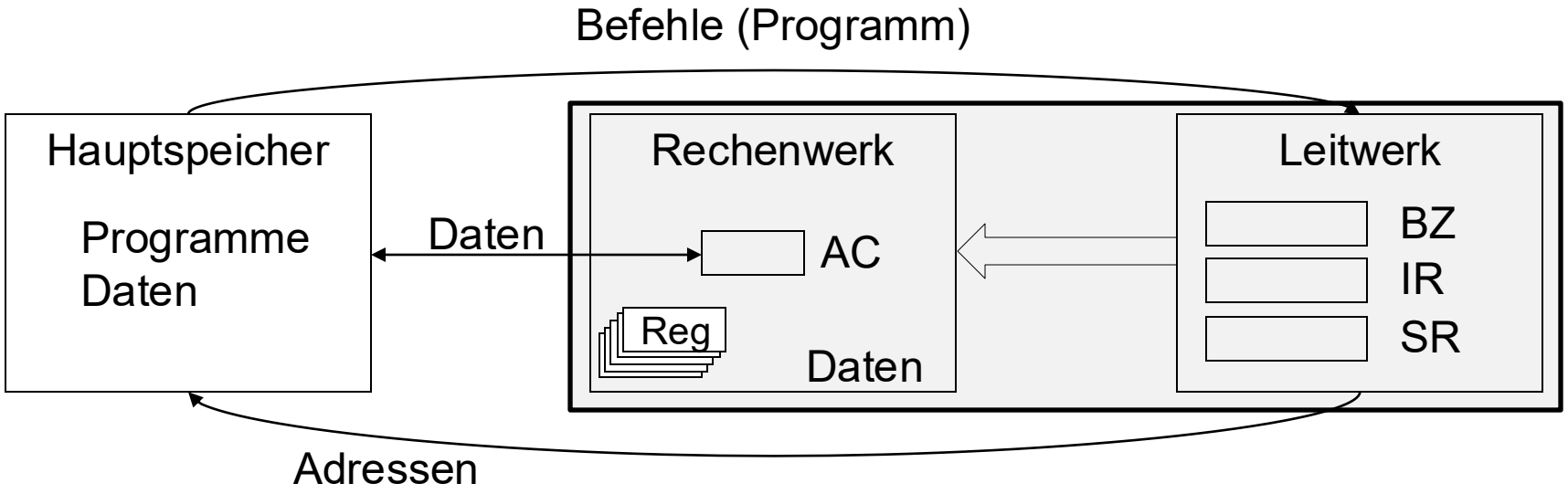
# Von-Neumann Architekturkonzept



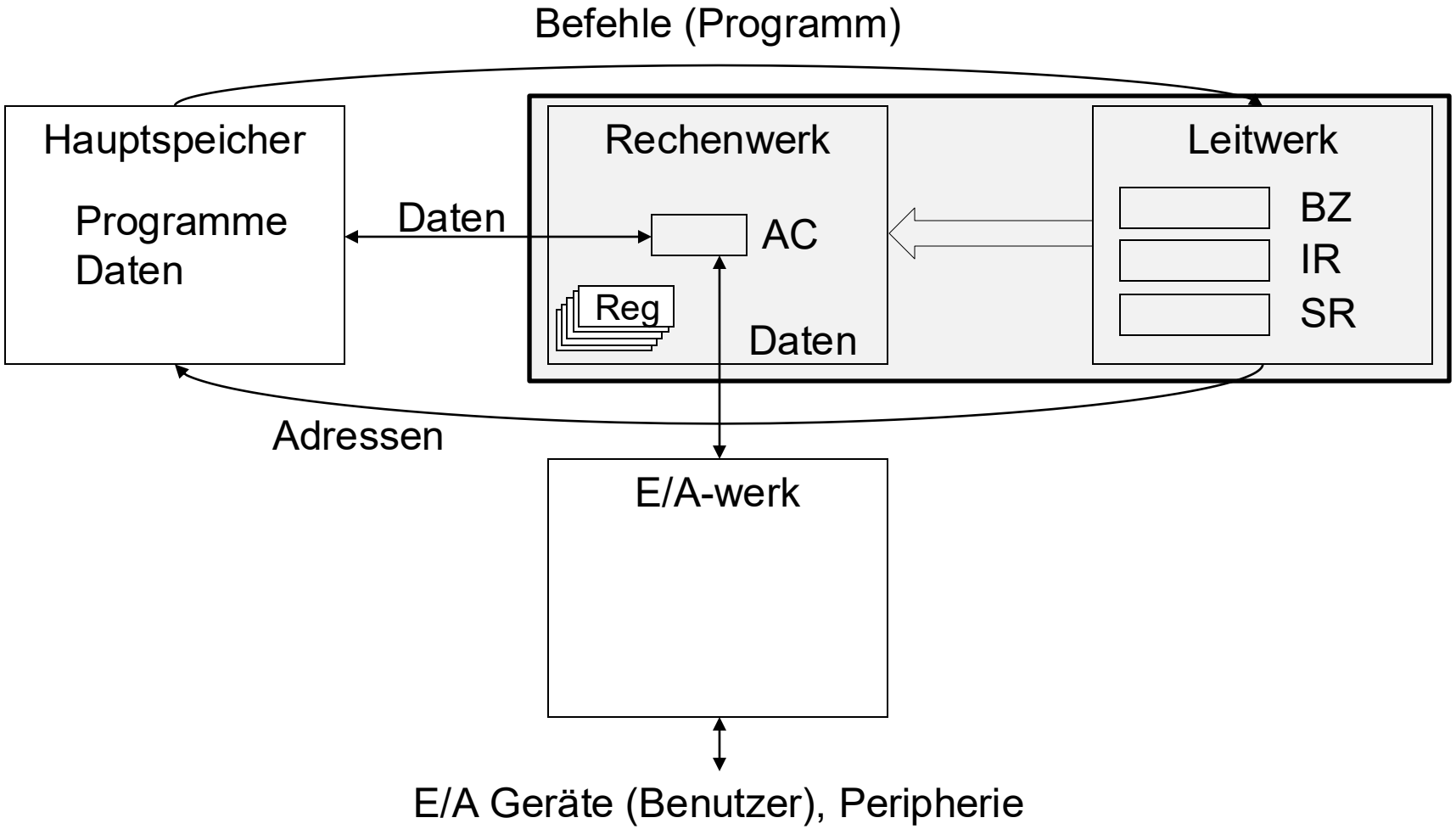
# Von-Neumann Architekturkonzept

1. Die Struktur des Rechners ist unabhängig vom bearbeiteten Problem (Programmsteuerung!)
2. Rechner besteht aus vier Werken
3. Der Hauptspeicher ist in Zellen gleicher Größe geteilt, die durch fortlaufende Nummern (Adressen) bezeichnet werden
4. Programm und Daten stehen in demselben Speicher (Hauptspeicher) und können durch die Maschine verändert werden
5. Die Maschine benutzt Binärcodes; Zahlen werden dual dargestellt
6. Das Programm besteht aus einer Folge von Befehlen (Instruktionen)
   - I.a. nacheinander gespeichert (aufsteigende Adressen)
   - I.a. nacheinander ausgeführt (sequentiell)
7. Von der Folge kann durch bedingte oder unbedingte Sprungbefehle abgewichen werden (Programmfortsetzung aus einer anderen Zelle)

# Von-Neumann Architekturkonzept mit Register



# Von-Neumann Architekturkonzept mit E/A Werk



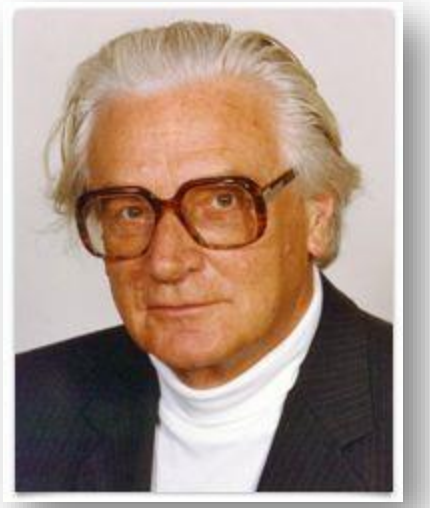
# Von-Neumann Architekturkonzept

1. Die Struktur des Rechners ist unabhängig vom bearbeiteten Problem (Programmsteuerung!)
2. Rechner besteht aus vier Werken
3. Der Hauptspeicher ist in Zellen gleicher Größe geteilt, die durch fortlaufende Nummern (Adressen) bezeichnet werden
4. Programm und Daten stehen in demselben Speicher (Hauptspeicher) und können durch die Maschine verändert werden
5. Die Maschine benutzt Binärcodes; Zahlen werden dual dargestellt
6. Das Programm besteht aus einer Folge von Befehlen (Instruktionen)
   - I.a. nacheinander gespeichert (aufsteigende Adressen)
   - I.a. nacheinander ausgeführt (sequentiell)
7. Von der Folge kann durch bedingte oder unbedingte Sprungbefehle abgewichen werden (Programmfortsetzung aus einer anderen Zelle)

# Vergleichbare Konzepte

In Deutschland wurden vergleichbare Konzepte durch Konrad Zuse entwickelt (unabhängig von v.-Neumann)

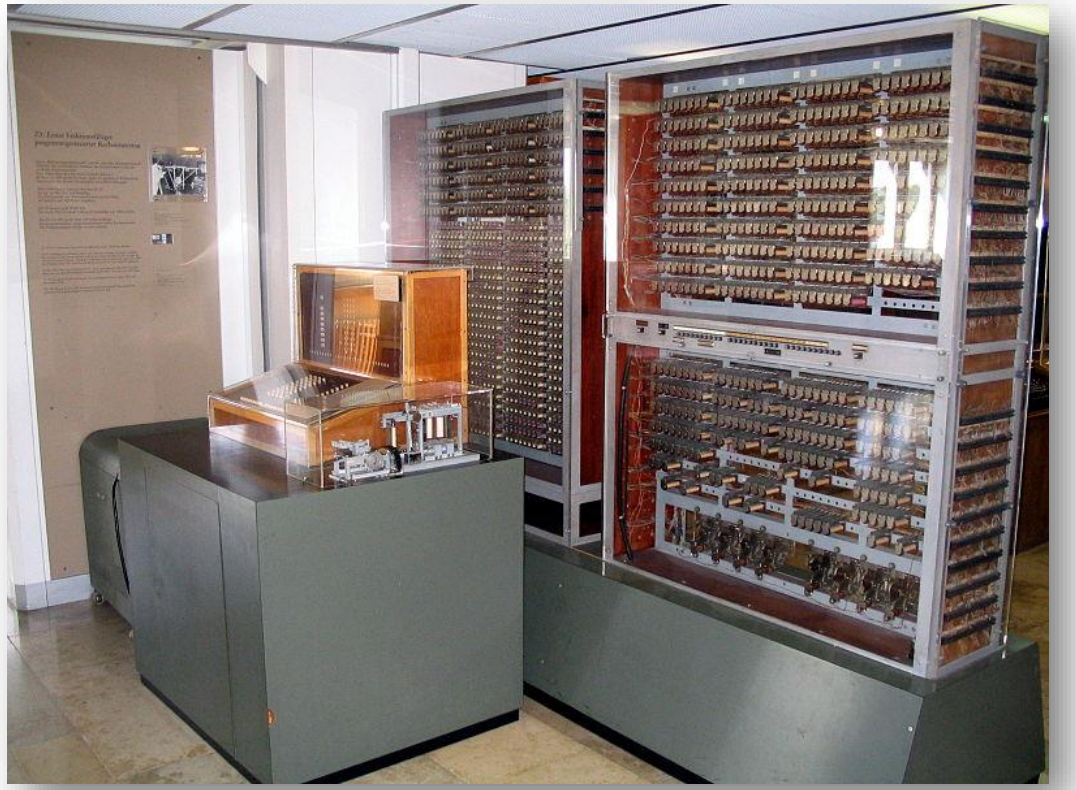
- Bau-Ingenieur
- Erfinder



Konrad Zuse
1910-1995

Z1: Mechanischer Rechner

- Programmgesteuert
- Technisches Museum in Berlin



Z3 bereits 1941(!) funktionsfähig

- Ausgestellt im Deutschen Museum (sowie eine Z4)
- Basierend auf Relais
- Keine bedingten Sprünge
- Programme nicht im Speicher

Funktionsfähige Z23 in Erlangen