A RISC Processor with Extended Forwarding

Gert Markwardt (Siemens AG)
Günter Kemnitz (TU Clausthal)
Reiner G. Spallek (TU Dresden)

Gliederung:

1. Motivation
2. Stackmaschine, Forwarding und erweitertes Forwarding
3. Befehlsformate
4. Experimentelle Ergebnisse zur Codedichte
5. Codegenerierung für Verzweigungen, Schleifen und Unterprogrammen

1 Motivation

a) Minimierung des Befehlsdatenstroms

b) Beobachtung, daß die meisten Zwischenergebnisse eines RISC-Programms nur eine kurze Lebensdauer besitzen

C-Programm

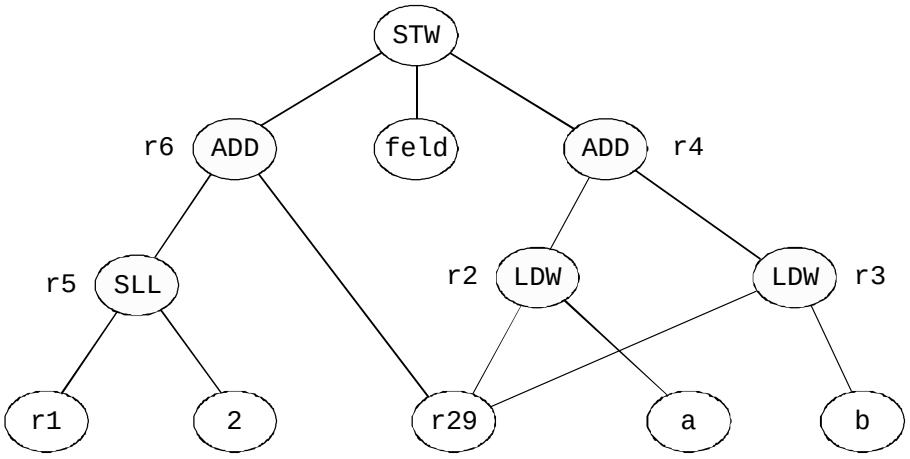
```
int *feld;
int i, a, b;
feld[i] = a + b;
```

RISC-Maschinenprogramm

```
LDW r2, r29, a     ;r2 ← M[r29 + a]
LDW r3, r29, b     ;r3 ← M[r29 + b]
ADD r4, r2, r3     ;r4 ← r2 + r3
SLL r5, r1, 2      ;r5 ← r1 * 4
ADD r6, r5, r29    ;r6 ← r5 + r29
STW r6, feld, r4   ;M[r6 + feld] ← r4
```

2 Stackmaschine, Forwarding und erweitertes Forwarding

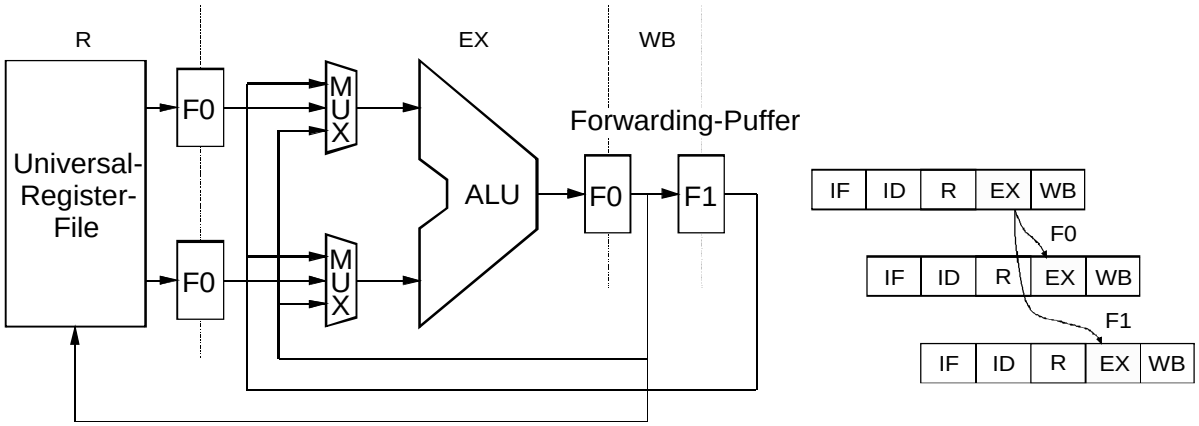
Verarbeitungsgraph



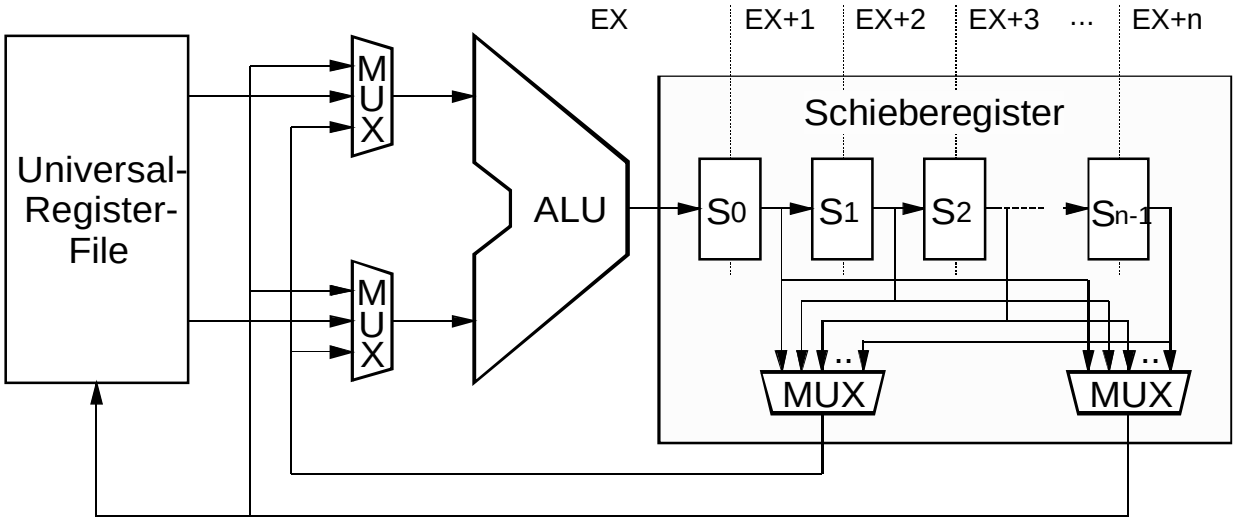
Abarbeitung auf einer Stackmaschine

		Inhalt des Stacks				
		0	-1	-2	-3	-4
PUSH	b	b				
PUSH	r29	r29	b			
LDW		r3				
PUSH	a	a	r3			
PUSH	r29	r29	a	r3		
LDW		r2	r3			
ADD		r4				
PUSH	feld	feld	r4			
PUSH	r29	r29	feld	r4		
PUSH	2	2	r29	feld	r4	
PUSH	r1	r1	2	r29	feld	r4
SLL		r5	r29	feld	r4	
ADD		r6	feld	r4		
STW						

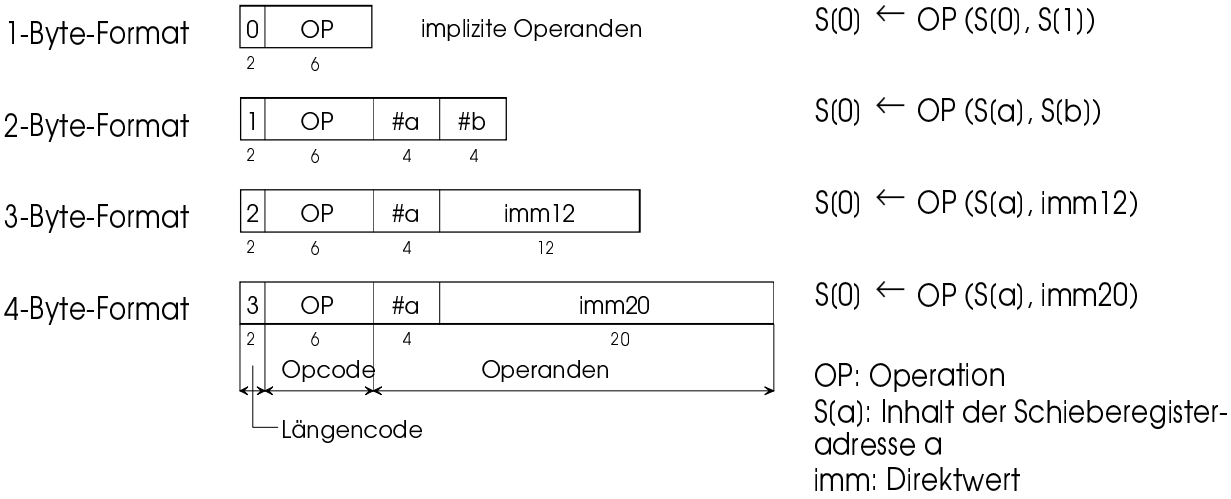
Forwarding von ALU-Ergebnissen



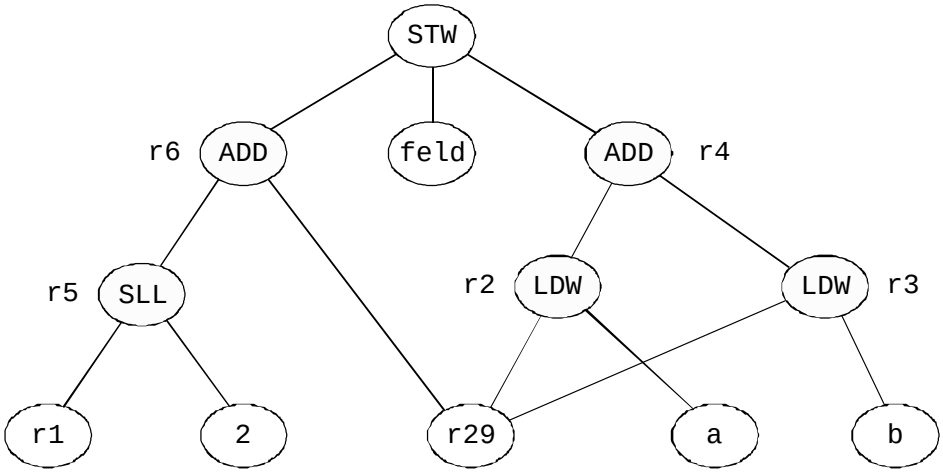
Erweitertes Forwarding



3 Befehlsformate für den Prozessor



Ein Maschinenprogramm




```
int *feld;
int i, a, b;
feld[i] = a + b;
```

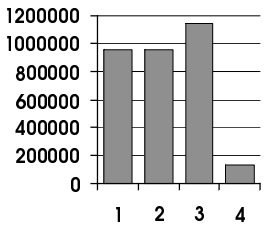
Inhalt des Schieberegisters

	#0	#1	#2	#3	#4
LDW r29, b	M[r29 + b]				
LDW r29, a	M[r29 + a]	M[r29+b]			
ADD	r4	M[r29+a]	M[r29+b]		
SLL r1, 2	r5	r4	M[r29+a]	M[r29+b]	
ADD #0, r29	r6	r5	r4	M[r29+a]	M[r29+b]
STW #0,feld, #2	r6	r5	r4	M[r29+a]	M[r29+b]

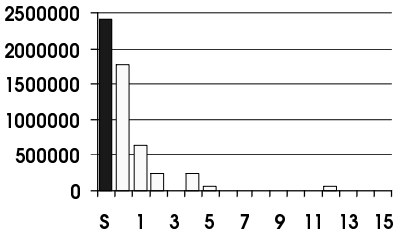
4 Experimentelle Ergebnisse

Bubble

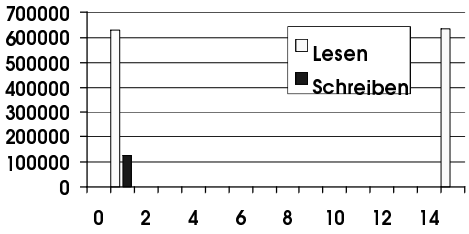
Befehlslängenverteilung



Schieberegisterzugriffe

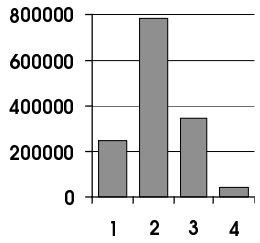


Universalregisterzugriffe

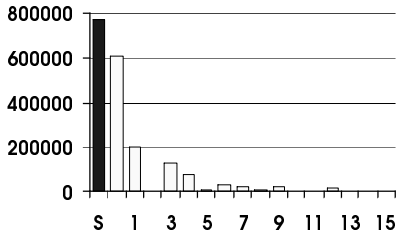


Quick

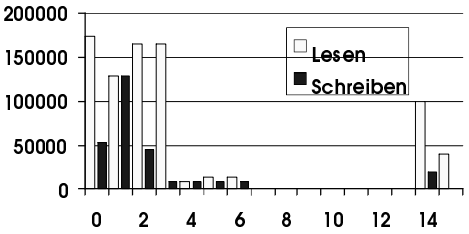
Befehlslängenverteilung



Schieberegisterzugriffe

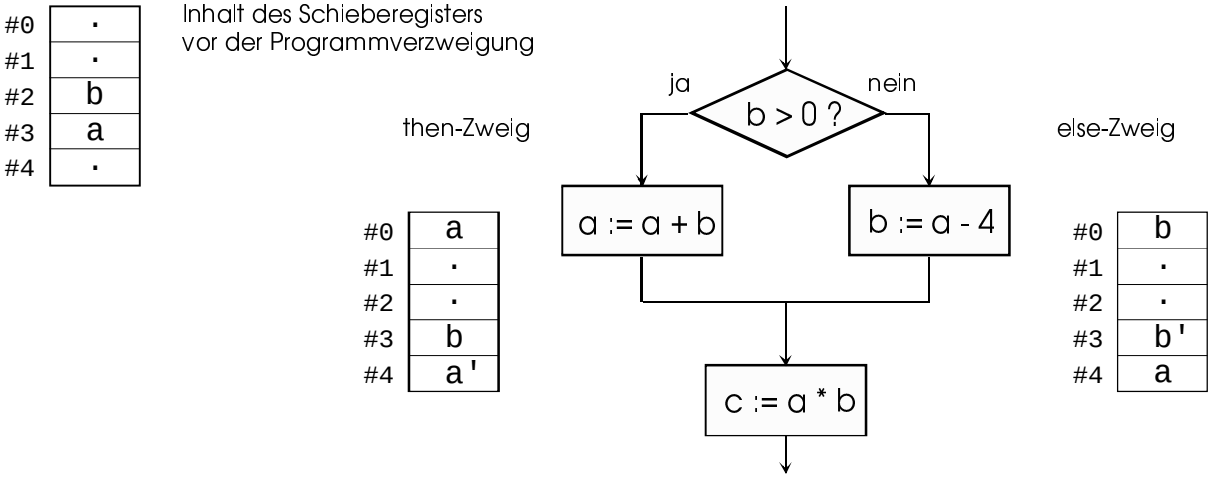


Universalregisterzugriffe

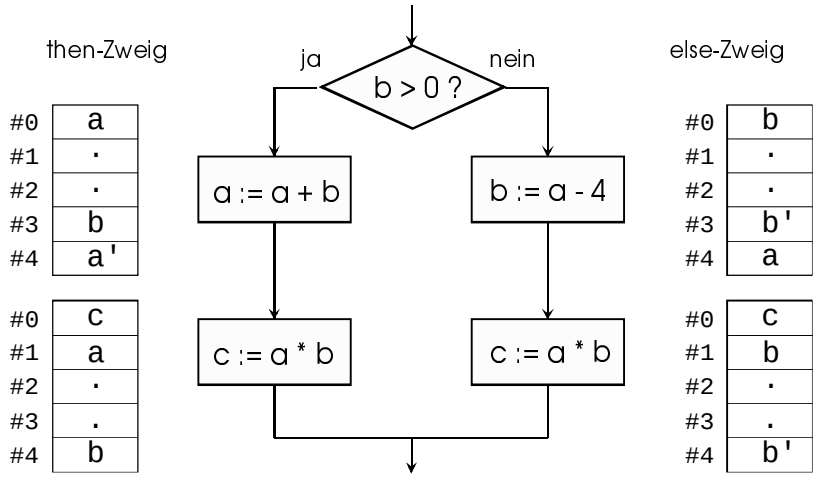


5 Codegenerierung für Verzweigungen, Schleifen und Unterprogramme

```
                              CMP #2, 0       ;if (b > 0)
      if (b > 0)              BLE L1
         a = a + b;           ADD #3, #2           ;then-Zweig
      else                    B   L2
         b = a - 4;     L1:   SUB #3, 4  ;else-Zweig
c = a * b;                   L2:   MUL #?, #? ;Schieberegisterinhalt nicht
                                   eindeutig
```



```
                        CMP  #2, 0          ;if
     if (b > 0)         BLE  L1
          a = a +       ADD  #3, #2    ;then-Zweig
b;                      MUL  #0, #3    ;eingefügte
          c = a * Multiplikation
b;                      B    L2
     else               L1:  SUB  #3, 4          ;else-Zweig
          b = a -       MUL  #4, #0    ;eingefügte
4;                      Multiplikation
          c = a * b;    L2:  ...
```



Zusammenfassung

- Ablegen der Ergebnisse in einem Schieberegister mit direkten Lesezugriff
- Adressierung durch den Befehlsabstand
- herkömmliches Register für Daten mit langer Lebensdauer
- Architekturvereinfachung
- Verbesserung der Codedichte
- Besonderheiten bei der Codegenerierung im Zusammenhang mit Verzweigungen, Schleifen, ...