h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi
FACHBEREICH INFORMATIK

RECHNERARCHITEKTUR

SS2024

Termin 2

Umgang Befehlssatz eines MU1 Prozessors

Name, Vorname	Matrikelnummer	Anmerkungen
Datum	Raster (z.B. Mi3x)	Testat/Datum

Legende: V:Vorbereitung, D: Durchführung, P: Protokoll/Dokumentation, T: Testat

Vorbereitung

Bereiten Sie die Lösungen daheim oder in den offenen Laboren so vor, dass Sie die Ergebnisse zum Labortermin möglichst präsentieren können.

Aufgabe1:

Erweitern sie den Befehlssatz des MU1 Prozessors um die Befehle PUSH, POP, LDR S, STR S, MOV PC und MOV SP. Zeichnen Sie in die Diagramme den jeweiligen Datenfluss und füllen Sie die Steuerungstabelle aus.

Der Befehl PUSH dekrementiert (SP=SP-1) den Stackpointer (Register SP) und speichert den aktuellen Akkumulatorinhalt (Register A) auf dem Stack.

Der Befehl POP lädt den Wert auf den der Stackpointer zeigt in den Akkumulator und inkrementiert (SP=SP+1) den Stackpointer.

Der Befehl STR S schreibt den Inhalt des Akkumulator in die Speicherstelle mit der Adresse, welche in der Speicherstelle mit der Adresse S steht.

Der Befehl LDR S lädt den Inhalt der Speicherstelle mit der Adresse, welche in der Speicherstelle mit der Adresse S steht, in den Akkumulator.

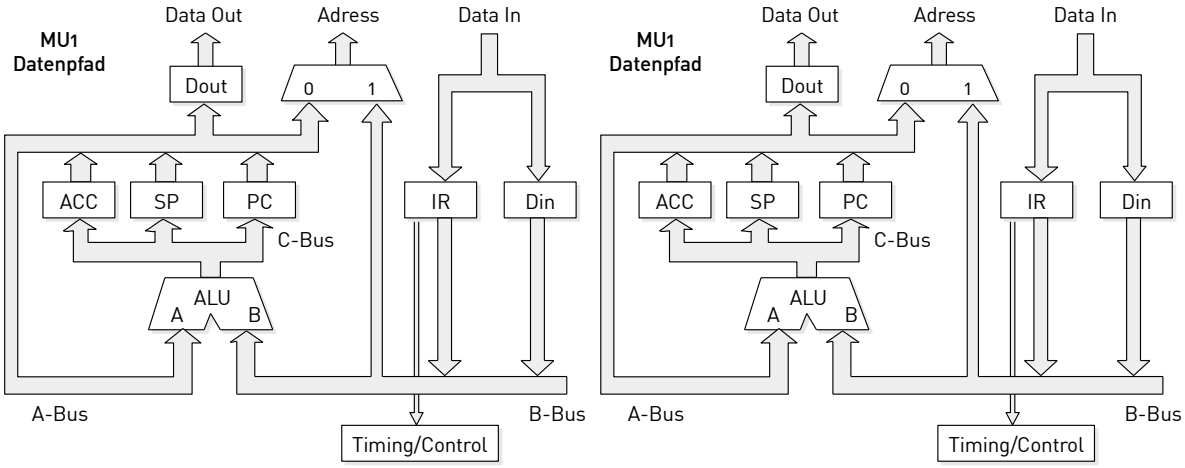
Der Befehl MOV PC kopiert den Inhalt vom Register ACC in das Register PC.

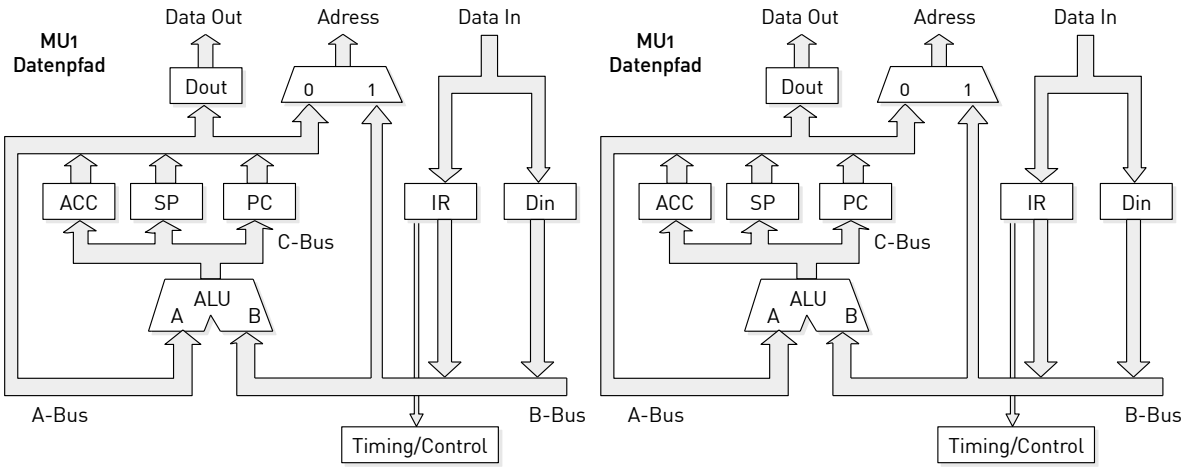
Der Befehl MOV SP kopiert den Inhalt vom Register ACC in das Register SP.

Befehlstabelle für MU1

Instruction	*Effekt*
Reset	PC = 0
LDA S	ACC = [S]
STO S	[S] = ACC
ADD S	ACC = ACC + [S]
JUMP S	PC = S
JGE S	IF ACC >= 0 PC = S
JNE S	IF ACC != 0 PC = S
STOP	stop
CALL S	SP = SP-1, [SP] = PC, PC = S
RETURN	PC = [SP], SP = SP + 1
PUSH	SP = SP-1, [SP] = ACC
POP	ACC = [SP], SP = SP + 1
LDR S	ACC = [[S]]
STR S	[[S]] = ACC
MOV PC	PC = ACC
MOV SP	SP = ACC

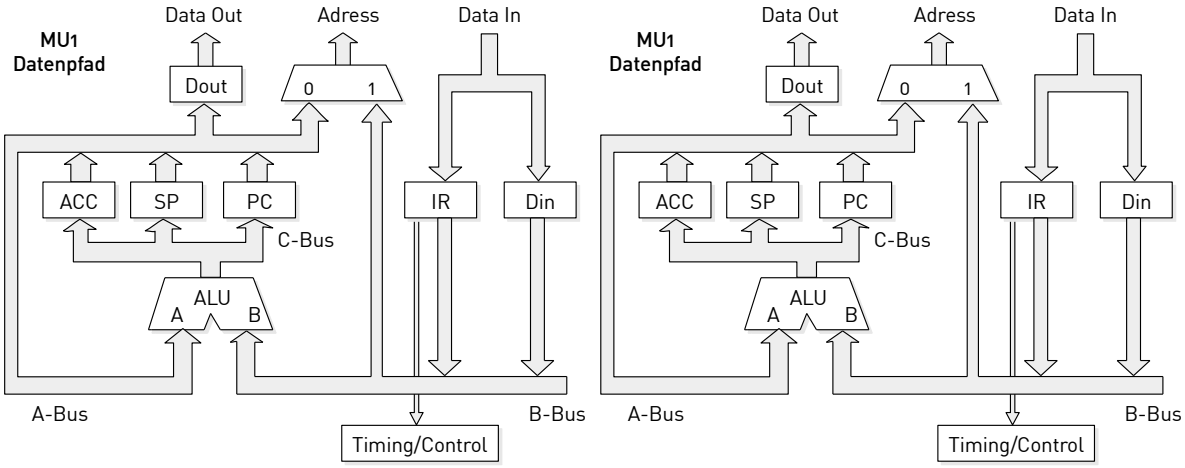
Der Befehl Push

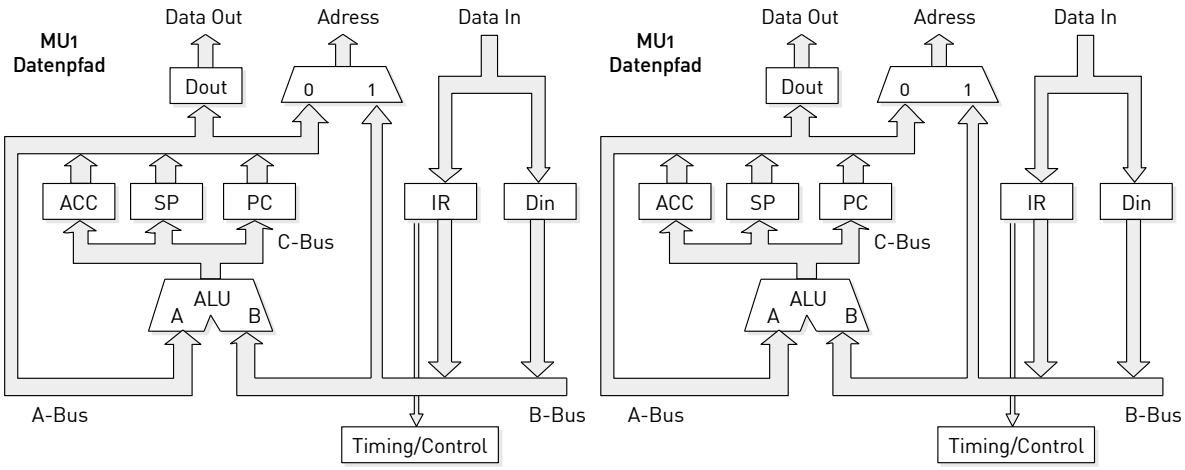




Inputs						Outputs																Description/Effect
																			Memory			
Instruction	Opcode	Reset	Step	ACC_Z / Zero	ACC_{15} / Negativ	Step	ACC_{OE}	ACC_{ie}	SP_{oe}	SP_{ie}	PC_{oe}	PC_{ie}	IR_{oe}	IR_{ie}	DIN_{oe}	DIN_{ie}	$DOUT_{oe}$	$DOUT_{ie}$	ALU	Adress	MEM_{rq}	R/W	
PUSH	1010_b = 0xA																					

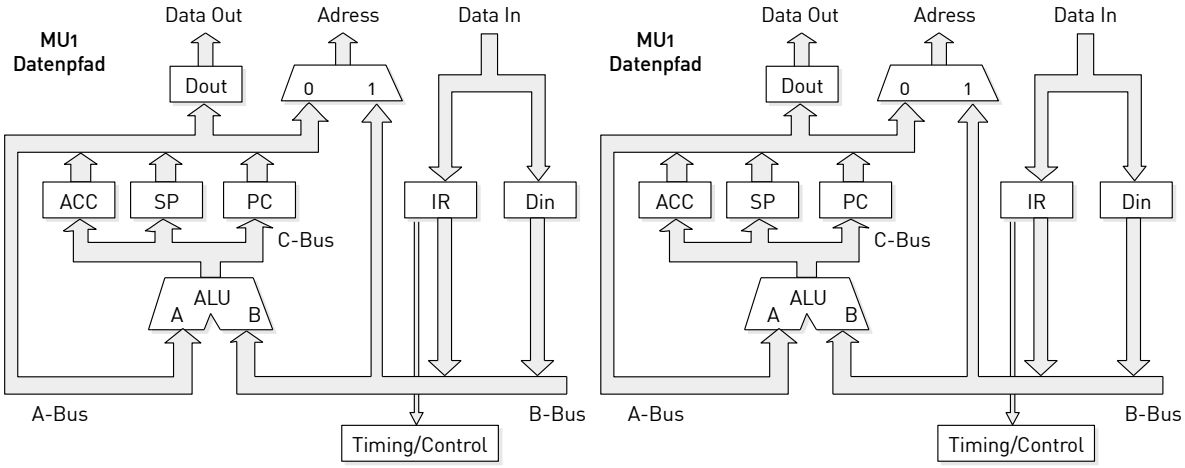
Der Befehl Pop

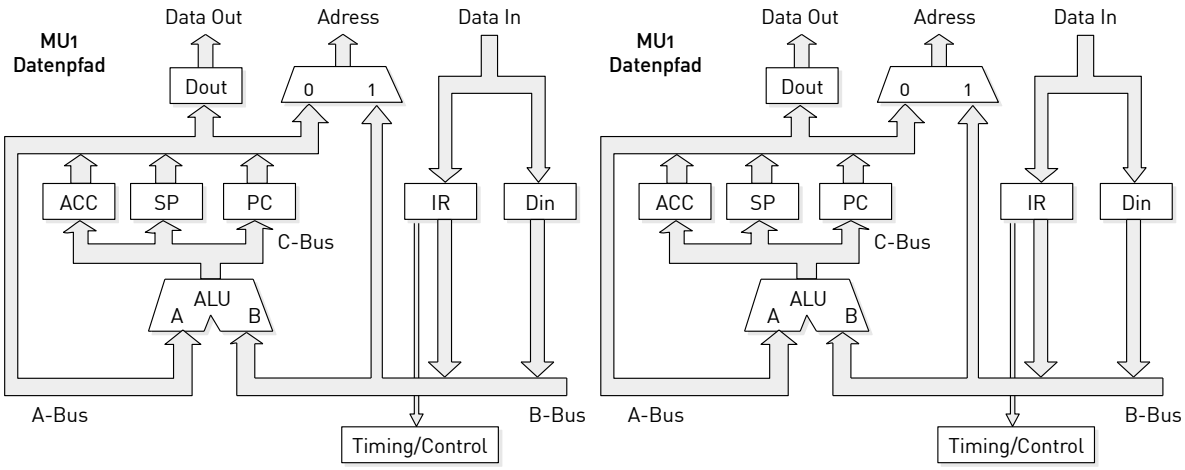




Inputs						Outputs																Description/Effect
																			Memory			
Instruction	Opcode	Reset	Step	ACC_Z / Zero	ACC_{15} / Negativ	Step	ACC_{OE}	ACC_{ie}	SP_{oe}	SP_{ie}	PC_{oe}	PC_{ie}	IR_{oe}	IR_{ie}	DIN_{oe}	DIN_{ie}	$DOUT_{oe}$	$DOUT_{ie}$	ALU	Adress	MEM_{rq}	R/W	
POP																							

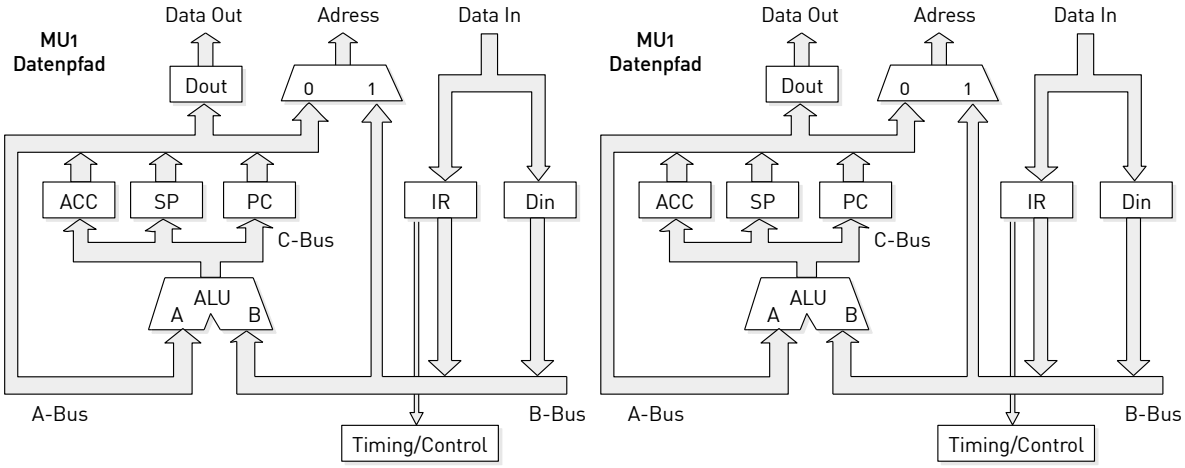
Der LDR S Befehl

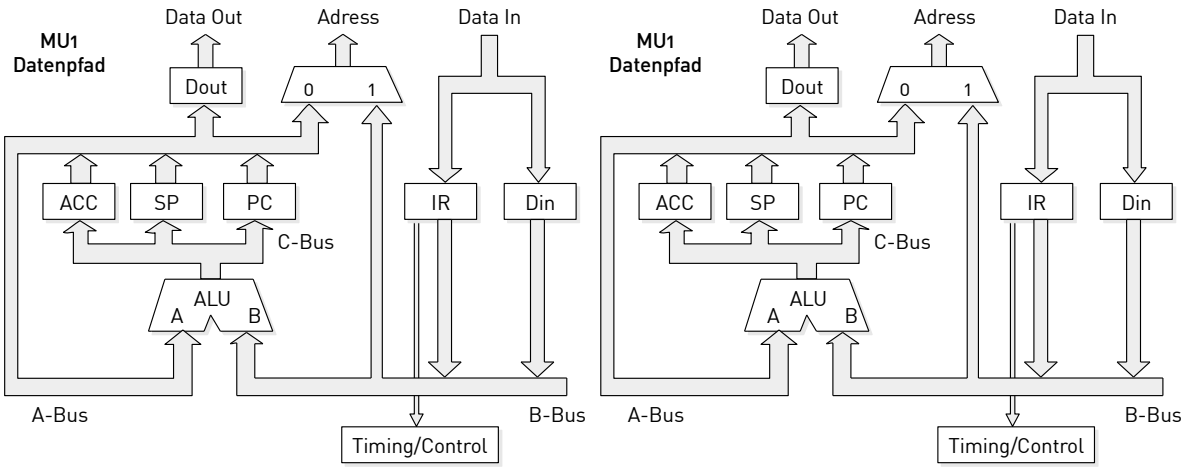




Inputs						Outputs																Description/Effect
																			Memory			
Instruction	Opcode	Reset	Step	ACC_Z / Zero	ACC_{15} / Negativ	Step	ACC_{OE}	ACC_{ie}	SP_{oe}	SP_{ie}	PC_{oe}	PC_{ie}	IR_{oe}	IR_{ie}	DIN_{oe}	DIN_{ie}	$DOUT_{oe}$	$DOUT_{ie}$	ALU	Adress	MEM_{rq}	R/W	
LDR S																						

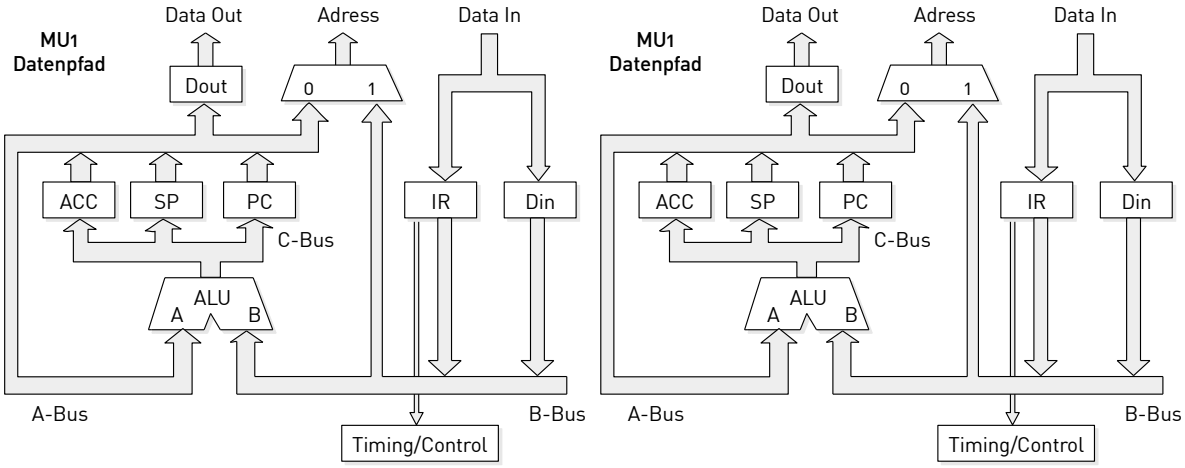
Der STR S Befehl

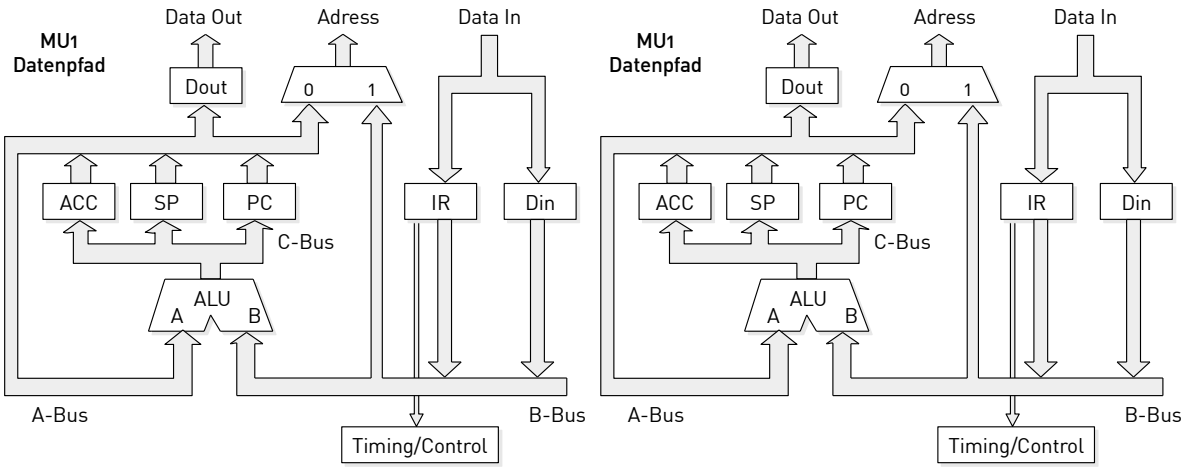




Inputs						Outputs																Description/Effect
																			Memory			
Instruction	Opcode	Reset	Step	ACC_Z / Zero	ACC_{15} / Negativ	Step	ACC_{OE}	ACC_{ie}	SP_{oe}	SP_{ie}	PC_{oe}	PC_{ie}	IR_{oe}	IR_{ie}	DIN_{oe}	DIN_{ie}	$DOUT_{oe}$	$DOUT_{ie}$	ALU	Adress	MEM_{rq}	R/W	
STR S																						

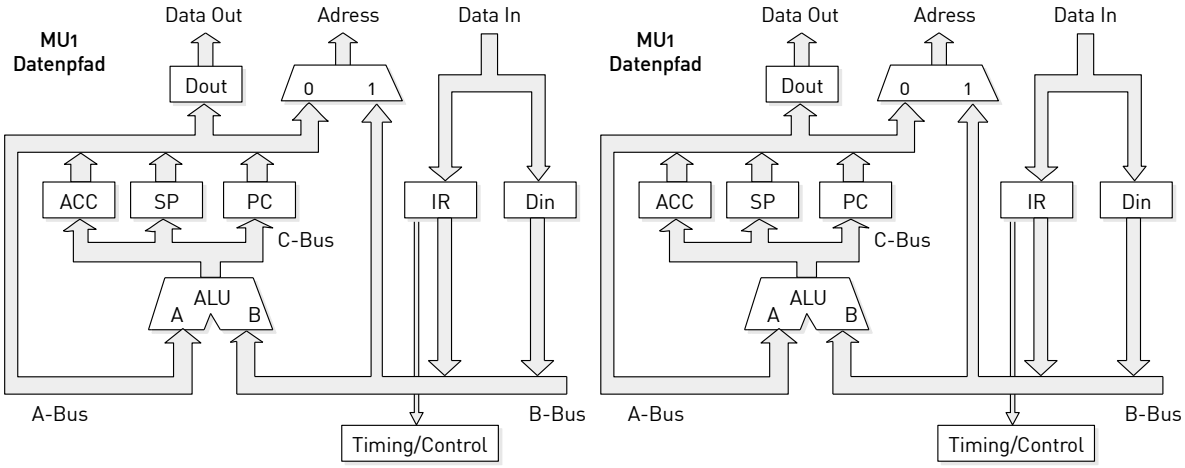
Der MOV PC Befehl

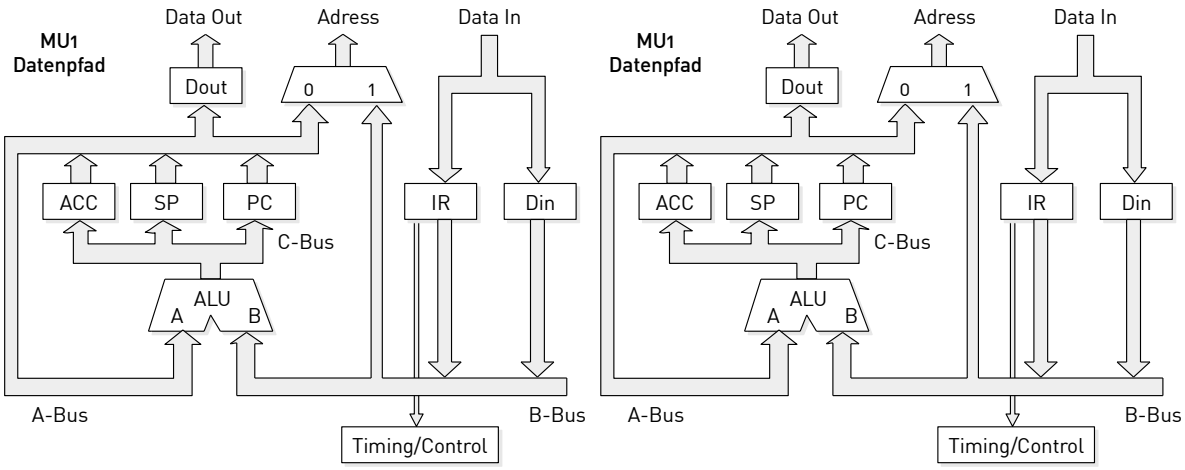




Inputs						Outputs																Description/Effect
																			Memory			
Instruction	Opcode	Reset	Step	ACC_Z / Zero	ACC_{15} / Negativ	Step	ACC_{OE}	ACC_{ie}	SP_{oe}	SP_{ie}	PC_{oe}	PC_{ie}	IR_{oe}	IR_{ie}	DIN_{oe}	DIN_{ie}	$DOUT_{oe}$	$DOUT_{ie}$	ALU	Adress	MEM_{rq}	R/W	
MOV PC																						

Der MOV SP Befehl





Inputs						Outputs																Description/Effect
																			Memory			
Instruction	Opcode	Reset	Step	ACC_Z / Zero	ACC_{15} / Negativ	Step	ACC_{OE}	ACC_{ie}	SP_{oe}	SP_{ie}	PC_{oe}	PC_{ie}	IR_{oe}	IR_{ie}	DIN_{oe}	DIN_{ie}	$DOUT_{oe}$	$DOUT_{ie}$	ALU	Adress	MEM_{rq}	R/W	
MOV SP																						

Aufgabe2:

Versuchen sie das Beispielprogramm aus der Vorlesung mit den neuen Befehlen LDR S und STR S so umzuschreiben, dass sie keinen selbst modifizierenden Code mehr benötigen.

```
Loop:       LDA   Total       ; Accumulate total
Add_instr:  ADD   Table       ; Begin at head of table
            STO   Total       ;
            LDA   Add_instr   ; Change address ...
            ADD   One         ; by modifying instruction!
            STO   Add_instr   ;
            LDA   Count       ; Count iterations
            SUB   One         ; Count down to zero
            STO   Count       ;
            JGE   Loop        ; If >= 0 repeat
            STP               ; Halt execution

; Data definitions
Total       DEFW  0       ; Total - initially zero
One         DEFW  1       ; The number one
Count       DEFW  4       ; Loop counter (loop 5x)
Table       DEFW  39      ; The numbers to total ...
            DEFW  25      ;
            DEFW  4       ;
            DEFW  98      ;
            DEFW  17      ;
```

Diese Aufgabe wird evtl. in einem der weiteren Termine (evtl. Termin 5) nochmals behandelt werden.