

Embedded Systems: Addressierung und Befehle



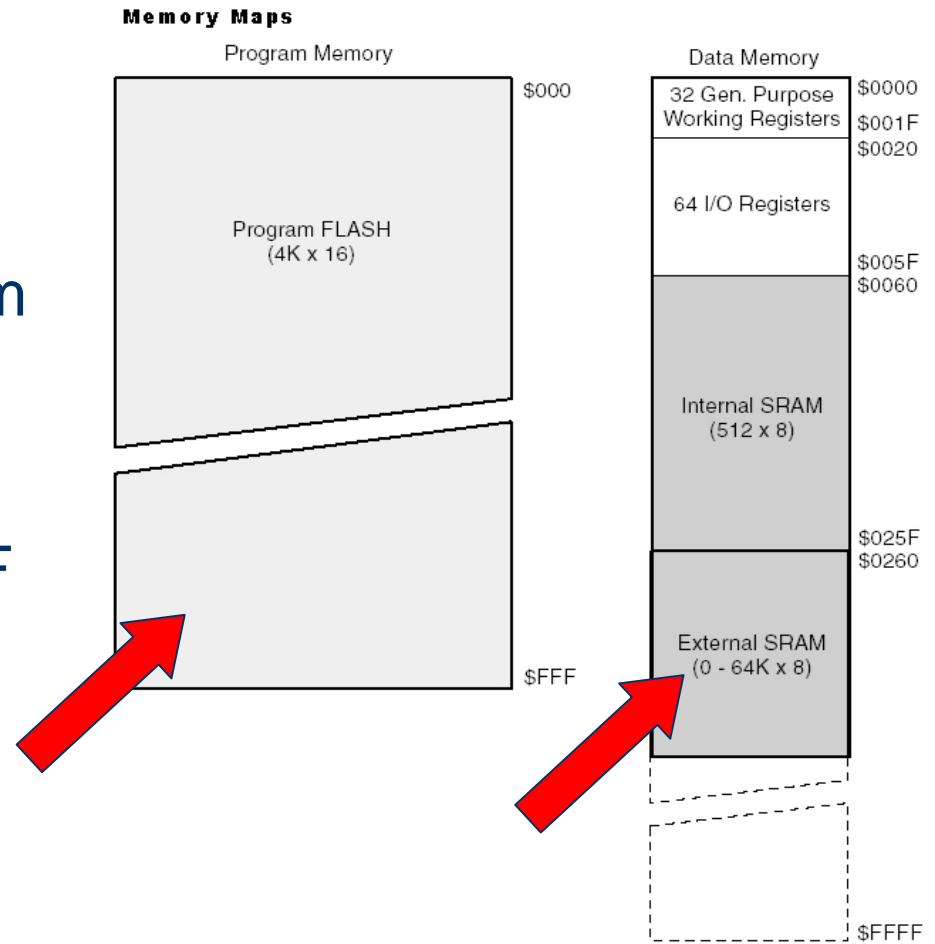
Memory Map

Programm Speicher:

- Adresse \$000-\$FFF
max. 4096 Byte für
kompiliertes Programm

Data Memory:

- Adresse \$0000-\$FFFF
 - 32 8Bit Register
 - 64 I/O Register
 - 512 Byte SRAM
 - 0-64 kByte Ext.SRAM



Befehle

Transportbefehle (unmittelbares schreiben):

- **LDI Destination(R16-R31), Value (8Bit)**

- LDI R17,\$AA

→R17 = \$AA

- **MOV Destination, Source (Reg. 0-31)**

- LDI R17,\$AA

- MOV R6,R17

→R17 = R6 = \$AA

Befehle

Arithmetik

- **ADD Destination, Source (R0-R31)**

- LDI R17,\$01
- LDI R18,\$02
- ADD R17,R18

→ R17 = \$03

- **SUB Destination, Source (R0-R31)**

- LDI R18,\$02
- LDI R17,\$01
- SUB R18,R17

→ R18 = \$01

- **MUL Destination, Source (R0-R31)**

- LDI R17,\$02
- LDI R18,\$96=150d
- MUL R18,R17
- Ergebnis hat 16Bit
R1=High Byte
R0=Low Byte
Ergebnis=\$012C

→ R1 = \$01 R0 = \$2C



Befehle

Logik

- **AND Destination, Source**

- LDI R17,\$04
- LDI R18,\$02
- AND R18,R17

→ R18 = \$00

- **OR Destination, Source**

- LDI R17,\$04
- LDI R18,\$02
- AND R18,R17

→ R18 = \$06

- **LSL Destination**

- LDI R17,\$04
- LSL R17

→ R17 = \$08
Jedes Bit wird eine Stelle nach links geschiftet

- **NEG Destination**

- LDI R17,\$04
- NEG R17

→ R17 = \$FC
(Zweierkomplement)

Befehle

Sprung- und Verzweigungsbefehle

- **RJMP Label**

- Springt an das angegebene Label

- **BRCS Label**

- LDI R17,\$FF
- LDI R18,\$05
- ADD R17,R18
- BRCS Carry_gesetzt

- **RCALL Label**

- Springt an angegebenes Label
- Dabei wird die Rücksprungadresse im Stack gespeichert
- Mit RET kann man zurückspringen



Adressregister

- X, Y und Z-Register (Adress-Register)
 - X-Register = R26 und R27
 - Y-Register = R28 und R29
 - Z-Register = R30 und R31
- Speicherraum 16Bit
 - Adressierung mit einem Register = 8 Bit nicht möglich
 - 2 Register zu einem Adressregister zusammenfassen

Speicherzugriff

Direkt Schreiben

- STS Adresse, Register
 - LDI R17,\$AF
 - STS \$0100,R17
- an Adresse \$0100 = \$AF

Direkt Lesen

- LDS Register, Adresse
 - LDS R18,\$0100
- R18 = \$AF



Speicherzugriff

Indirekt Schreiben

- ST Adressregister, Source
 - LDI R17,\$AF
 - LDI R26,\$60
 - SUB R27,R27
 - ST X,R17
- an Adresse \$60 = \$AF

Indirekt Lesen

- LD Desitination, Adressregister
 - LDI R17,\$44
 - STS \$60,R17
 - LDI R26,\$60
 - CLR R27
 - LD R18,X
- R18 = \$44

Speicherzugriff

Indirekt Indiziert Schreiben

- STD Adressregister, Source
 - LDI R17,\$AF
 - LDI R26,\$60
 - SUB R27,R27
 - STD X+2,R17
- an Adresse \$62 = \$AF

Indirekt Indiziert Lesen

- LDD Desitination, Adressregister
 - LDI R17,\$44
 - STS \$62,R17
 - LDI R26,\$60
 - CLR R27
 - LDD R18,X+2
- R18 = \$44



Übungen

Übungen

Aufgabe 1

- Schreiben sie ein Programm, welches die Zahlen 50,10 und 30 addiert. Benutzen Sie dazu nur die Register R16 und R17. Das Ergebnis sollte in Register R17 stehen. Danach sollte das Ergebnis von Register R17 in Register R6 gespeichert werden.

LDI	R16,50	R16 = 50
LDI	R17,10	R17 = 10
ADD	R16,r17	R16 = 60
LDI	R17,30	R17 = 30
ADD	R17,R16	R17 = 90
MOV	R6,R17	R6 = 90

Übungen

Aufgabe 2

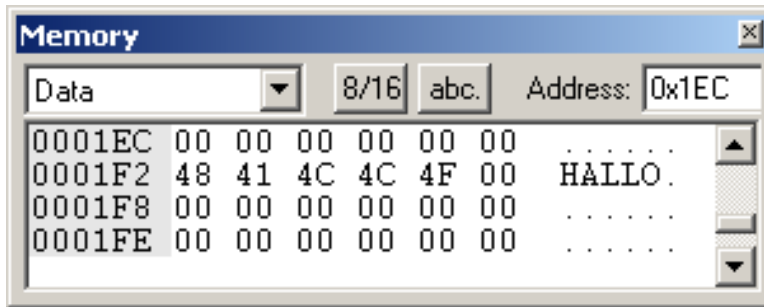
- Betrachten sie folgende Zeilen Assembler-code. Was steht nach Abarbeitung des Codes in R6.

LDI R16,3	R16 = 0011	
MOV R7,R16	R7 = 0011	
LSL R7	R7 = 0110	
MOV R6,R7	R6 = 0110	
LSL R6	R6 = 1100	1100
AND R6,R16	R6 = 0000	0000

Übungen

Aufgabe 3

- Betrachten sie folgenden Speicherauszug:



LDI R31,01

LDI R30,\$F3

LD R3,Z

- Schreiben Sie ein Programm, unter Verwendung des Z-Adressregisters (R31:R30), dass den Buchstaben „A“ von „HALLO“ aus dem Speicher in Register R3 liest.