Rechnerorganisation Übungsblatt 1

# Ü 1.1 Zweierkomplement

1. Subtraktionen werden zu simplen additionen. Einerkomplement erfordert spezielle operation für Sub
2. 1111 1111 1111 1111 = -1^10 = 177777^8 = 0xFFFF  
   0000 1001 0010 0111= 2343 ….  
   1100 1101 0010 1000 = -26040^10 = 146450^8 = 0xCD28
3. Ungerade: Daran dass die letzte Zahl ein 1er ist (wie immer Binär)  
   Negativ: Most significant bit ist eine 1

# Ü 1.2 Arithmetische und logische Operationen

1. 01001010 +  
   10100111 =  
   11110001 = -15
2. 00111001 -  
   10100011 =  
   11011100 = 36
3. 11010110 XOR  
   10110011 =   
   01100101 = 101
4. 01111100 >> 2 =  
   00011111 = 31
5. 11110111 AND  
   01011001 = 81
6. 11001110 NOT  
   00110001 = 49

# Ü 1.3 MIPS Grundlagen

1. Erklären Sie die übliche Speicheraufteilung der MIPS-Architektur. Wo werden lokale Variablen, globale Variablen bzw. der auszuführende Programmcode abgelegt?

Lokale Variablen: $t0-$t9  
Globale Variablen: $s0-$s9  
Programmcode: $ra (Return address) $sp (Stackpointer) $fp (Framepointer) $at

1. Wie viele „General Purpose Register“ stehen Ihnen in der MIPS-I ISA zur Verfügung? Nach welchem Schema (Konventionen) werden diese im Allgemeinen genützt?

$f0-$f31 -> General purpose  
Sobald mit Gleitkomma gerechnet wird benötigen wir 64 Bit deshalb werden immer gerade und ungerade zu einem Register zusammengenommen ($f0-f1… )  
$f0,$f1 Werden für die Benutzereingabe benutzt  
$f12/13 Wird für die Ausgabe benutzt

1. In der MIPS-I ISA werden Instruktionen als 32-Bit-Werte gespeichert. Was verstehen Sie in diesem Zusammenhang unter I-, R- bzw. J-Format-Instruktionen? Erklären Sie diese drei Instruktionsformate anhand von drei Befehlen Ihrer Wahl.

R = Operation mit 2 Registern. Z.B.: add $rd, $rs, $rt => $rd = $rs + $rt  
I = Immediate Operationen mit einem Register und einem numerischen wert. Z.B.:  
addi $rd, $rs, imm => $rd = $rs + imm  
J = Jump Befehl. 26Bit sind für die Zieladresse reserviert z.B.: j dest => PC = address \* 4

1. Wie können folgende Instruktionen im Maschinencode dargestellt werden: sub $s0, $t1, $t2; addi $t0, $t1, 10; sll $s1, $s2, 3; bzw. li $s0, 0x34203404

**addi $t0, $t1 10:**  
0010 0010 0000 1001 0000 0000 0000 1010 = OP: 8; $t0 = 8; $t1 = 9; 10 imm = 10  
**sll $s1, $s2, 3:**0000 0010 0011 0010 0000 0000 0011 0000 = OP: 0; $s1 = 17; $s2 = 18; shamt = 3; func = 0 **li $s0, 0x34203404:**LI = ori $s0, $zero, 0x34203404 =  
0011 0110 0010 0000 1101 0000 1000 0000 1101 0000 0001