Projektbericht

Instruction	Instruction Encoding
EOR (shifted register)	f100 1010 ss0m mmmm iiii iinn nnnd dddd
ADDS (immediate)	f011 0001 0hii iiii iiii iinn nnnd dddd
SUBS (immediate)	f111 0001 0hii iiii iiii iinn nnnd dddd
B.cond	0101 0100 iiii iiii iiii iiii iii0 cccc
ADDS (shifted register)	f010 1011 ss0m mmmm iiii iinn nnnd dddd
RET	1101 0110 0101 1111 0000 00nn nnn0 0000

Assembly Language	Machine Code (binary)	Machine Code (hex)
eor x0, x0, x0	1100 1010 0000 0000 0000 0000 0000 0000	ca000000
eor x1, x1, x1	1100 1010 0000 0001 0000 0000 0010 0001	ca010021
eor x2, x2, x2	1100 1010 0000 0010 0000 0000 0100 0010	ca020042
adds x0, x0, #5	1011 0001 0000 0000 0001 0100 0000 0000	b1001400
adds x1, x1, #3	1011 0001 0000 0000 0000 1100 0010 0001	b1000c21
adds x2, x2, #7	1011 0001 0000 0000 0001 1100 0100 0010	b1001c42
subs x0, x0, #1	1111 0001 0000 0000 0000 0100 0000 0000	f1000400
b.ne my_loop	0101 0100 1111 1111 1111 1111 1010 0001	54ffffa1
adds x0, x1, x2	1010 1011 0000 0010 0000 0000 0010 0000	ab020020
ret	1101 0110 0101 1111 0000 0011 1100 0000	d65f03c0

Aufgabe 10.2

bne_0 – bne_3:
Von bne_0 bis bne_3 wird gesprungen. Das erkennen wir daran, dass der Program Counter auf 0x400010 steht. Dies steht für die nächste Codezeile die ausgeführt wird: nämlich die 5. Zeile (Index 4 => 4*4 = 16 = 0x10).
Es bleibt die Frage, wie die 0x10 in den Program Counter (PC) kommt. Die Antwort darauf beginnt in der Instruction Memory. Der dort beschriebene Offset wird in Sign Extend weitergeleitet. Sign Extend verlängert den Offset von einer 19 auf eine 64-Bit Zahl. Dies ist nötig, da ein Einfaches hinzufügen von Nullen den Zahlenwert im Zweierkomplement ändert. Außerdem hat der Program Counter eine Länge von 64 Bit, wodurch diese Operation unerlässlich wird.
Vor dem Addieren auf den derzeitigen Program Counter, wird der Offset um 2 Bits nach links geshifted. Dies entspricht einer Multiplikation mit 4 und muss getan werden, da jede Operation 4 Bytes lang ist. Nach dem Addieren liegt am Multiplexer die korrekte Sprungadresse an.
Das von unten eingehende Eingangsbit vom Multiplexer gibt an, ob wir zur Sprungadresse springen oder um eine Instruktion weitergehen.
Von bne_0 bis bne_3 ist dieses Bit 1, da es ein Branchbefehl ist und die Branchbedingung wahr (Z=0) ist.
Das wird festgestellt, da die Flagbranch aktiviert. Diese wird mit dem dazugehörigen Flag verUNDet (hier das Zeroflag).
bne_4:
Am Multiplexer liegt unten eine 0 an, da die Branchbedingung falsch (Z=1) ist. Deshalb wird auf dem Program Counter 4 hinzuaddiert und wir gehen eine Instruktion weiter. Am Program Counter sieht man, dass wir zu 0x400020 gehen.