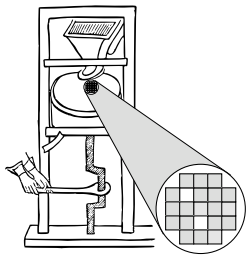
Technische Universität Darmstadt
FG Eingebettete Systeme und
ihre Anwendungen (ESA)

Prof. Dr. Andreas Koch
Holger Lange
Mathias Halbach (FG Rechnerarchitektur)



13.07.2006

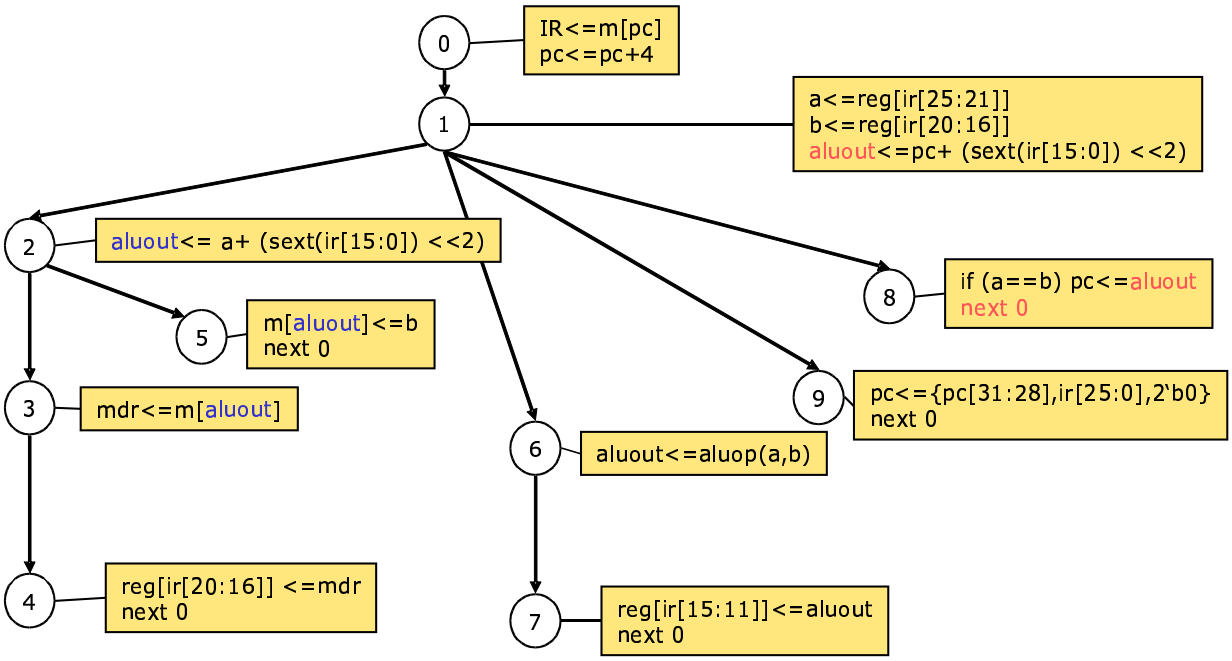
Technische Grundlagen der Informatik II
10. Übung – MIPS Pipelining
Sommersemester 2006

Aufgabe 1: Taktdauer

a) Wie lange dauert jeweils die Ausführung der Befehle **`load`**, **`store`**, **`add`**, **`beq`** und **`jump`** bei der Mehrtakt-Implementierung mit einem variabel langem Takt (so kurz wie möglich für den aktuellen Befehl), wenn folgende Zeiten vorgegeben sind:

$t_{regread} = 1$ ns
$t_{regwrite} = 1$ ns
$t_{ALU} = 2$ ns
$t_{readmem} = 3$ ns
$t_{writemem} = 3$ ns

Alle anderen Zeiten werden zu Null angenommen ($t_{cto} = 0, t_{setup} = 0, t_{mux} = 0, t_{and} = 0, t_{control} = 0$, usw.). Für jede Mikrooperation ist die minimale Taktperiode anzugeben und im folgenden Zustandsdiagramm einzutragen.

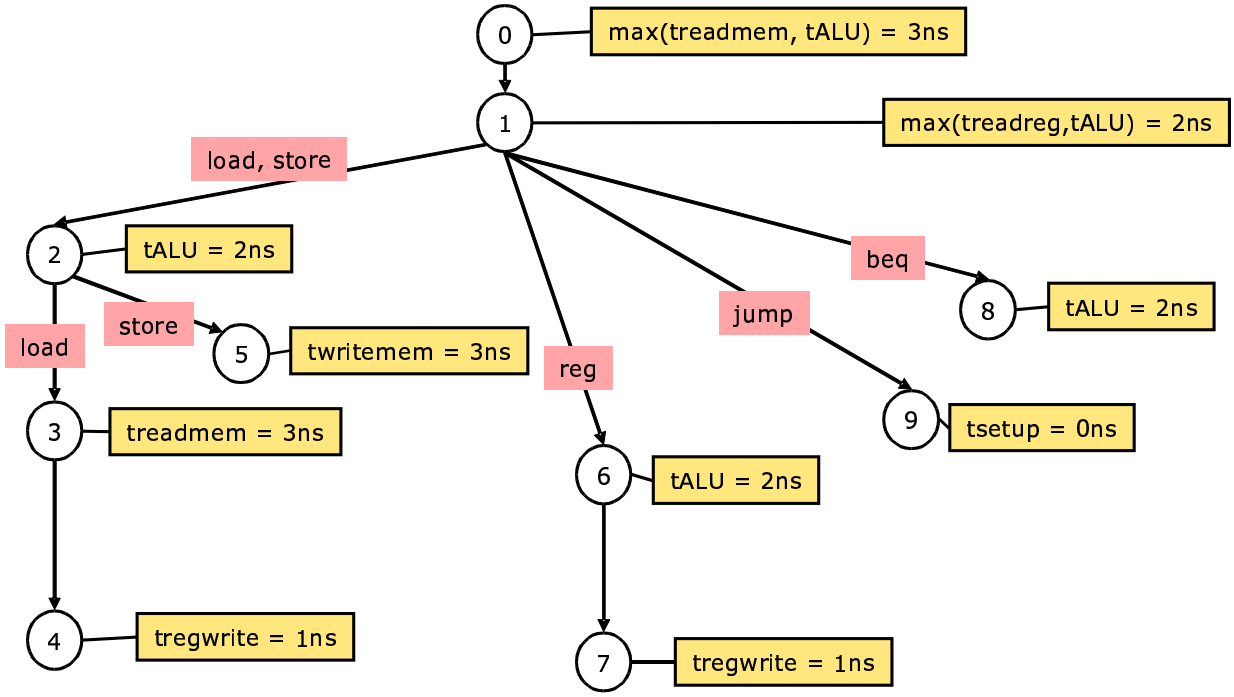


b) Geben Sie nun für jeden Befehl die Summe der Zeiten für die jeweils auszuführenden Mikrooperationen an.

c) Mit welchem Takt würden Sie eine MIPS-Pipeline bei den obigen Zeiten betreiben?

Lösung:

a)



b)

load:	3 ns + 2 ns + 2 ns + 3 ns + 1 ns = 11 ns
store:	3 ns + 2 ns + 2 ns + 3 ns = 10 ns
add:	3 ns + 2 ns + 2 ns + 1 ns = 8 ns
beq:	3 ns + 2 ns + 2 ns = 7 ns
jump:	3 ns + 2 ns + 0 ns = 5 ns

c) $t_{readPmem} = 3$ ns ist die minimal benötigte Taktdauer. Daraus ergibt sich eine maximale Taktfrequenz von 333,33 MHz.

Aufgabe 2: Pipelining

Lassen Sie für eine MIPS-Pipeline ohne Forwarding die Befehlsfolge

```
add  $2, $1, $3
sub  $12, $2, $5
add  $14, $2, $2
lw   $15, 100($2)
```

durchlaufen, wobei die Register $i am Anfang jeweils den Wert i besitzen. Schreiben Sie für jede einzelne Stufe (und je Befehl) in zeitlicher Abfolge auf, welche Werte für die nächste Stufe benötigt werden. Für einen Befehl lw $6, 5($4) wäre dies z.B.:

IF ← lw $6, 5($4)
ID ← (lw, $6, 5, reg[4] = 4)
EX ← (lw, $6, 5 + 4 = 9)
MEM ← ($6, mem[9])
WB: reg[6] := mem[9]

Stellen Sie fest, welche Werte sich danach in den Zielregistern ergeben. Was müssen Sie ändern, um die gleichen Ergebnisse wie ohne Pipelining zu erhalten? Schreiben Sie dazu den geänderten Ablauf und die Ergebniswerte der Register auf.

Lösung:

Die einzelnen Stufen der Pipeline IF (Instruction Fetch), ID (Instruction Decode), EX (Execution), MEM (Memory) und WB (Write Back) je Befehl untereinander dargestellt ergeben die Liste:

Takt	Befehl 1	Befehl 2	Befehl 3	Befehl 4
1	IF ← add $2, $1, $3			
2	ID ← (add, $2, reg[1] = 1, reg[3] = 3)	IF ← sub $12, $2, $5		
3	EX ← ($2, 1 + 3 = 4)	ID ← (sub, $12, reg[2] = 2, reg[5] = 5)	IF ← add $14, $2, $2	
4	MEM ← ($2, 4)	EX ← ($12, 2 - 5 = -3)	ID ← (add, $14, reg[2] = 2, reg[2] = 2)	IF ← lw $15, 100($2)
5	WB: reg[2] := 4	MEM ← ($12, -3)	EX ← ($14, 2 + 2 = 4)	ID ← (lw, $15, 100, reg[2] = 4)
6		WB: reg[12] := -3	MEM ← ($14, 4)	EX ← (lw, $15, 100 + 4 = 104)
7			WB: reg[14] := 4	MEM ← ($15, mem[104])
8				WB: reg[15] := mem[104]

Aus obiger Liste lassen sich folgende Registerwerte ablesen:

Register	Wert
2	4
12	-3
14	4
15	Speicherinhalt (32 Bit) der Adresse 104

Um die gleichen Ergebnisse wie ohne Pipelining zu erhalten, müssen nach dem ersten Befehl zwei NOP-Befehle eingefügt werden. Daraus ergibt sich der Ablauf:

Takt	Befehl 1	Befehle 2-4	Befehl 5	Befehl 6
1	IF ← add $2, $1, $3			
2	ID ← (add, $2, reg[1] = 1, reg[3] = 3)	(IF ← nop)		
3	EX ← ($2, 1 + 3 = 4)	(IF ← nop)		
4	MEM ← ($2, 4)	IF ← sub $12, $2, $5		
5	WB: reg[2] := 4	ID ← (sub, $12, reg[2] = 4, reg[5] = 5)	IF ← add $14, $2, $2	
6		EX ← ($12, 4 - 5 = -1)	ID ← (add, $14, reg[2] = 4, reg[2] = 4)	IF ← lw $15, 100($2)
7		MEM ← ($12, -1)	EX ← ($14, 4 + 4 = 8)	ID ← (lw, $15, 100, reg[2] = 4)
8		WB: reg[12] := -1	MEM ← ($14, 8)	EX ← (lw, $15, 100 + 4 = 104)
9			WB: reg[14] := 8	MEM ← ($15, mem[104])
10				WB: reg[15] := mem[104]

Die nun korrekten Registerwerte sind:

Register	Wert
2	4
12	-1
14	8
15	Speicherinhalt (32 Bit) der Adresse 104

Aufgabe 3: Pipeline in Verilog

Implementieren Sie die Berechnung von $y = (a+b)-(c+(d-3))$

a) als kombinatorisches Schaltnetz

b) als 3-stufige Pipeline

in Verilog. Alle Werte sind 32 Bit breit.

Lösung:

a)

```
module combinational(a, b, c, d, y);
  input [31:0] a, b, c, d;
  output [31:0] y;

  wire [31:0] y;
  assign y = (a + b) - (c + (d - 3));

endmodule
```

oder

```
module combinational(a, b, c, d, y);
  input [31:0] a, b, c, d;
  output [31:0] y;

  reg [31:0] y;

  always @(a or b or c or d)
    y = (a + b) - (c + (d - 3));

endmodule
```

b)

```
module pipeline(clk, reset, a, b, c, d, y);
  input clk, reset;
  input [31:0] a, b, c, d;
  output [31:0] y;

  reg [31:0] y, sum_a_b, diff_d_3, sum_c_diff;

  always @(posedge clk or posedge reset) begin
    if (reset) begin
      y <= 0;
```

```
      sum_c_diff <= 0;
      diff_d_3 <= 0;
      sum_a_b <= 0;
    end
    else begin
      y <= sum_a_b - sum_c_diff;
      diff_d_3 <= d - 3;
      sum_c_diff <= c + diff_d_3;
      sum_a_b <= a + b;
    end
  end

endmodule
```