


27-Nov-06 (1)




# CSC2510 - Computer Organization

## Lecture 13: Revision Processor Unit

Philip Leong

27-Nov-06 (2)




## Execution of Add (R3),R1

- Fetch the instruction
- Fetch the first operand (R3)
- Perform the addition
- Load result to R1

- What should MDRin be?

27-Nov-06 (3)




## Execution

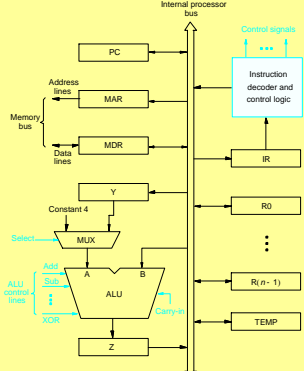
- Step 1: PC loaded into MAR, read request to memory, MUX gives 4, added to B (PC) and stored in Z
- Step 2: Z moved to PC while waiting for memory
- Step 3: Word fetched from memory and loaded into IR
- Step 4: figure out what the instruction should do and set control circuitry for steps 4-7. R3 transferred to MAR and memory read operation initiated
- Step 5: contents of R1 moved to Y
- Step 6: read operation completed and is in MDR as well as B input of ALU. Select Y as second input of ALU and add performed
- Step 7: result is transferred to R1, End causes a goto step 1

- Which steps are the instruction fetch?

27-Nov-06 (4)



## Execution



Step	Action
1	PC <sub>out</sub> , MAR <sub>in</sub> , Read, Select4, Add, Z <sub>in</sub>
2	Z <sub>out</sub> , PC <sub>in</sub> , Y <sub>in</sub> , WMF C
3	MDR <sub>out</sub> , IR <sub>in</sub>
4	R3 <sub>out</sub> , MAR <sub>in</sub> , Read
5	R1 <sub>out</sub> , Y <sub>in</sub> , WMF C
6	MDR <sub>out</sub> , SelectY, Add, Z <sub>in</sub>
7	Z <sub>out</sub> , R1 <sub>in</sub> , End

Figure 7.6. Control sequencor execution of the instruction Add (R3),R1.

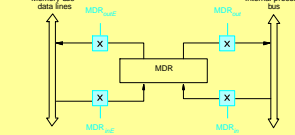



Figure 7.4. Connection and control signals for MDR.

27-Nov-06 (5)




## Branch Instructions

Step	Action
1	PC <sub>out</sub> , MAR <sub>in</sub> , Read, Select4, Add, Z <sub>in</sub>
2	Z <sub>out</sub> , PC <sub>in</sub> , Y <sub>in</sub> , WMF C
3	MDR <sub>out</sub> , IR <sub>in</sub>
4	Offset-field-of-IR <sub>out</sub> , Add, Z <sub>in</sub>
5	Z <sub>out</sub> , PC <sub>in</sub> , End

- Steps 1-3, instruction fetch
- Step 4: add the offset to the PC
- Step 5: update the PC

- Now do you understand why the branch offset is calculated from the next address to be executed?
- For conditional e.g. branch < 0, step 4 is replaced with
- Offset-field-of-IR<sub>out</sub>, Add, Z<sub>in</sub>, If N=0 then End

27-Nov-06 (6)



## Multiple Buses

- One disadvantage of our single bus scheme is that only one data item can be transferred over the bus per cycle
- A solution is multiple internal buses
- All registers combined into a register file with 3 ports
  - Why are there 2 outputs?
  - What is the input for?
  - What does 3 port mean?
- Buses A and B allow simultaneous transfer of the two operands for the ALU
  - ALU is able to just pass one of its operands to R e.g. R=A
- Incrementer unit computes PC+4, means we don't need the ALU for this
  - ALU still has a 4 input for other instructions such as postincrement

### Three bus datapath

- What does this do?
- (WMFC means wait for memory function completed)
- What are the advantages and disadvantages over a single bus?

---

**Step Action**

1	$PC_{out}$ , $R=B$ , $MAR_{in}$ , Read, IncPC
2	WMFC
3	$MDR_{outB}$ , $R=B$ , $IR_{in}$
4	$R4_{outA}$ , $R5_{outB}$ , SelectA, Add, $R6_{in}$ , End

### Hardwired Control

- How do we generate the control signals?
  - Hardwired control
  - Microprogrammed control
- A hardwired control is called a **finite state machine**
  - Sequences using a counter and produces control signals at the right time
  - Control signals are functions of the IR, external inputs and condition codes
  - Can you give an example for each?

### Microprogrammed Control

- The control signals are stored in a memory as sequences of **control words** which are the individual bits of the control signals
- **Microinstructions** are executed in a manner similar to machine code

Micro - instruction	PC <sub>in</sub>	PC <sub>out</sub>	MAR <sub>in</sub>	Read	MDR <sub>out</sub>	IR <sub>in</sub>	Y <sub>in</sub>	Select	Add	Z <sub>in</sub>	Z <sub>out</sub>	R <sub>1</sub> <sub>out</sub>	R <sub>1</sub> <sub>in</sub>	R <sub>3</sub> <sub>out</sub>	WMFC	End
1	0	1	1	1	0	0	0	1	1	1	0	0	0	0	0	0
2	1	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0
3	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
4	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0
5	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0
6	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1

### Microprogrammed Control Unit

- A **microprogram counter** is used to read control words sequentially from control store
- Every time new instruction loaded into the IR, output of "Starting Address Generator" loaded into the uPC
- uPC automatically incremented by clock causing successive microinstructions to be read from the control store
- Control signals delivered to various parts of the processor in the correct sequence
- This scheme is not able to change its sequence as a result of other inputs such as the condition code e.g. Branch < 0

Figure 7.16. Basic organization of a microprogrammed control

### Microprogrammed Control Unit

Address	Microinstruction
0	$PC_{out}$ , $MAR_{in}$ , Read, Select4, Add, $Z_{in}$
1	$Z_{out}$ , $PC_{in}$ , $Y_{in}$ , WMFC
2	$MDR_{out}$ , $IR_{in}$
3	Branch to starting address of appropriate microroutine
25	If $N=0$ , then branch to microinstruction 0
26	Offset-field-of- $IR_{out}$ , SelectY, Add, $Z_{in}$
27	$Z_{out}$ , $PC_{in}$ , End

Figure 7.17. Microroutine for the instruction Branch<0.

### Scheme to allow Conditional Branching

- "Starting and branch address generator"
  - Loads new address into uPC when instructed
  - Has condition codes and external inputs which can affect uPC
- uPC incremented every cycle except
  - When new instruction loaded into IR, uPC loaded with starting address of the microroutine
  - For taken branches, uPC updated to branch address
  - For End microinstruction, uPC set to 0