CSE1400 - Computer Organisation

Self-Study: Week 6
Basic Processing Unit

Delft University of Technology

2021/2022 Q1

Special thanks to Sára Juhošová, Patrik Barták, Matej Havelka and Kiril Vasilev, Ana Cristiana Marcu for helping with the compilation of this set of questions.

Important information:

1. If any question is unclear please consult Stack Overflow. For more specific questions, you can use the Queue during lab hours.
2. The average time for solving this self study is **3** hours, and **1** hour is allocated to giving feedback. Timings are included for each exercise to give you a more clear overview of how much time you should be spending on them.
3. The maximum amount of points for this self study is 200 points. To get the points you should submit a serious attempt on Peer and **properly review** your peers' submissions (100 points per full cycle, including review evaluation).
4. Answers will be provided during the weekly tutorial sessions.

1 Instruction Execution

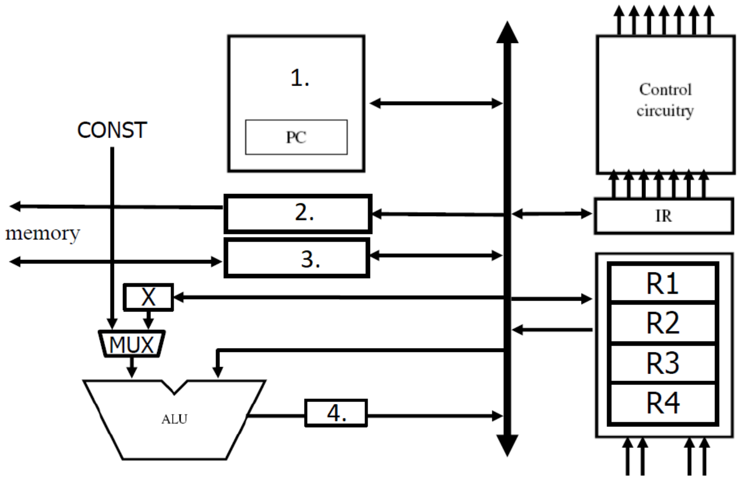


Figure 1: Basic Processing Unit with missing components.

1. (5 mins) Figure 1 shows the components of a CISC-style basic processing unit. Name the numbered components and describe what kind of information is stored in them.

#	name	description
1	instruction address generator	generates the address of the next instruction
2	MAR	Memory Address Register – stores memory address
3	MDR	Memory Data Register – stores data from memory
4	Z	stores the result of ALU operations

2. (5 mins) Write down the 5 stages of a RISC-based BPU and explain what is happening during each of them:

1. Fetch – get instruction from memory
2. Decode – evaluate the OP code and the encoded register addressing
3. Execute – perform arithmetic and logic operations in the ALU
4. Memory – access information from memory if needed
5. Writeback – communicate back to registers if needed

3. (10 mins) Consider the following assembly code:

```
          foo:
1.                  pushq %rbp
2.                  movq %rsp, %rbp
3.                  jmp bar
          foo2:
4.                  pushq %rbp
5.                  movq %rsp, %rbp
6.                  addq %r12, %r15
7.                  jmp end
          bar:
8.                  pushq %rbp
9.                  movq %rsp, %rbp
10.                 jmp foo2
          end:
11.                 movq $0, %rdi
12.                 call exit
```

Write down the value of the program counter (PC) for every instruction (during the instruction execution stage) until the code finishes. Assume that the line number corresponds to the memory location of the instruction on that line and the exit instruction is at memory location 64.

#	instruction being executed	value in PC
1	pushq %rbp	2
2	movq %rsp, %rbp	3
3	jmp bar	8
4	pushq %rbp	9
5	movq %rsp, %rbp	10
6	jmp foo2	4
7	pushq %rbp	5
8	movq %rsp, %rbp	6
9	addq %r12, %r15	7
10	jmp end	11
11	movq $0, %rdi	12
12	call exit	64

2 Control Signals

4. (5 mins) Write down the microroutine for fetching instructions. Assume that the PC is updated automatically by the Instruction Address generator.

1. PC_{out}, MAR_{IN}, READ, WMFC
2. MDR_{out}, IR_{IN}

We fetch the instruction from PC to MAR and then read it after which we fetch it to IR to be able to control circuity.

5. (8 mins) Create the most efficient microroutine for the `Add R2, (R4), R3` instruction after it has been fetched. Explain why do you think this is the most efficient one.

1. $R2_{out}$, X_{IN}
2. $R4_{out}$, MAR_{IN}, READ, WMFC
3. MDR_{out}, ADD, Z_{IN}
4. Z_{out}, $R3_{IN}$

4 step execution is the most optimal one, since we will always need 2 steps for decoding data, 1 step for the add operation to be performed in the ALU and 1 step for outputting the result.

6. (8 mins) Create a microroutine for the `Divide (R1), (R2), R3` instruction after it has already been fetched.

1. $R1_{out}$, MAR_{IN}, READ, WMFC
2. MDR_{out}, X_{IN}
3. $R2_{out}$, MAR_{IN}, READ, WMFC
4. MDR_{out}, DIVIDE, Z_{IN}
5. Z_{out}, $R3_{IN}$

7. (8 mins) Create the most efficient microroutine for the `BRANCH PC + (R3)` instruction after it has already been fetched. Assume that the memory address at R3 holds the offset for jumping.

1. $R3_{out}$, MAR_{IN}, READ, WMFC
2. MDR_{out}, X_{IN}
3. PC_{out}, X_{out}, ADD, Z_{IN}
4. Z_{out}, PC_{IN}

8. (10 mins) A classic BPU contains 16 registers and an ALU with 104 instructions which uses X as input and Z as output. How many bits do we need to cover all the control signals?

16 registers $= 2^4 \rightarrow 2(4+1) = 10$ bits
104 instructions $< 128 = 2^7 \rightarrow 7$ bits
$X \rightarrow$ only IN $\rightarrow 1$ bit
$Z \rightarrow$ IN and out $\rightarrow 2$ bits
$10+7+2+1 = 20$ bits

9. (10 mins) Below are three sequences of instruction stages. How long does each command take to execute? Which of these sequences executes the instruction R0 ← R1 + (R2) in the *least* amount of time? Every clock tick takes 25 ms and memory access takes 100 ms (assume no cache). Assume that both the bus and register Y are ALU inputs, and that accessing memory takes more than one clock cycle.
Note: 'R1' uses register addressing and '(R2)' uses register indirect addressing.

A		B		C	
$R2_{out}$,MAR_{in},READ,$\underline{\text{WMFC}}$ 100	25	$R1_{out}$,Y_{in}	25	$R2_{out}$,MAR_{in},READ $\curvearrowleft$	25
MDR_{out},Y_{in}	25	$R2_{out}$,MAR_{in},READ,$\underline{\text{WMFC}}$ 100	25	$R1_{out}$,Y_{in},WMFC	25
$R1_{out}$,Y_{out},ALU add,Z_{in}	25	MDR_{out},Y_{out},ALU add,Z_{in}	25	MDR_{out},Y_{out},ALU add,Z_{in}	25
Z_{out},$R0_{in}$	25	Z_{out},$R0_{in}$	25	Z_{out},$R0_{in}$	25
	200ms		200 ms		175 ms

Instruction set A first accesses memory which takes 100 ms and performs 3 25ms tasks. [175 ms]
B, on the other hand, first performs a task, then accesses memory and finishes off with 2 tasks [175 ms]. Finally, C accesses memory, while performing a task, and afterwards performs two more 25ms tasks, making it the fastest. [150 ms]

3 Hardwired Control Signals

10. (15 mins) A BPU has four instructions in total, two of which are `ADD` and `SUB`. The instructions are 8 bits long (x_0 to x_7) with the first 2 bits reserved for the opcode. The BPU uses hardwired control signals with a 3-bit (a_0, a_1, a_2) counter and has the following properties:

    - the opcode for the `ADD` instruction is `01`
    - the opcode for the `SUB` instruction is `11`
    - only the `ADD` and `SUB` instructions use the Z register
    - If the BPU does not need to communicate with the memory, the counter skips a stage (jumps by 2). For example, if the BPU needs to access memory after stage 2 (`01`) it goes to stage 3 (`10`) otherwise it jumps to stage 4 (`11`)

    Express the control signals for Z_{in} using boolean algebra.
    Hint: the stages order is Fetch, Decode, (Possible) Memory Access, Execute, Writeback.

	a_0	a_1	a_2
Fetch	0	0	0
Decode	0	0	1
Mem. Add.	0	1	0
$\boxed{\text{Execute}}$	0	1	1
Writeback	1	0	0

	$\overbrace{x_0 \quad x_1}^{OP}$		x_2	x_3	x_4	x_5
Add	0	1	...			
Sub	1	1	...			

$$\underbrace{\overline{a_0}\cdot a_1 \cdot a_2}\cdot x_1$$

$a_1 = a_2 = 1$

a_0 cannot be 1 as well

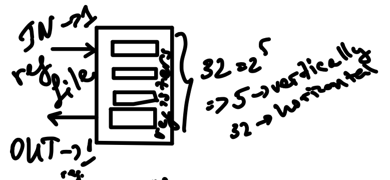
can remove $\overline{a_0}$

4 Microprogrammed Control Signals

11. (12 mins) Consider the following components of a BPU:

- A register file containing 32 registers
- The MAR and MDR registers
- An ALU capable of performing 20 operations

(a) Calculate the number of bits needed for the section of a vertically encoded micro instruction that controls these components.



$$\underbrace{2+5+5}_{\text{reg file}} + \underbrace{3}_{\text{MAR + MDR}} + \underbrace{5}_{\text{ALU}} = 20$$

$2 + 32 + 32 + 3 + 20 = 89$

32 registers $= 2^5 \to 2(5+1) = 12$ bits

MAR + MDR $= 1 + 2 = 3$ bits

20 ALU operations $\to 2^5 \to 5$ bits

$12 + 3 + 5 = 20$ bits

X register → 1 bit

2 register → 2 bits

MUX = 1 bit

4 bits + 20 bits = 24 bits

(b) Calculate the same for a horizontally encoded micro instruction.

32 registers $\to 2(32+1) = 66$ bits

MAR + MDR $= 1 + 2 = 3$ bits

20 ALU operations $= 20$ bits

$66 + 3 + 20 = 89$ bits

X register → 1 bit

2 register → 2 bits

MUX → 1 bit

89 bits + 4 bits = 93 bits

12. (15 mins) Consider the following simplified micro instruction set with mixed encoding. The register file has 16 registers (encoded as 4 bits) and the ALU's `ADD` instruction is encoded as `01101`. Registers are addressed based on their number (e.g. R5 = `0101`). The format of a micro instruction is given below, where each x represents one bit.

RF$_{\text{read}}$	**RF$_{\text{write}}$**	**RF$_{\text{addr-out}}$**	**RF$_{\text{addr-in}}$**	**F$_{\text{ALU}}$**	**X$_{\text{in}}$**	**MUX$_{\text{select(const/x)}}$**	**Z$_{\text{in}}$**	**Z$_{\text{out}}$**	**END**
x	x	xxxx	xxxx	xxxxx	x	x	x	x	x

01101 - ADD

(a) The BPU is performing the `ADD R4, R7, R8` operation (summing the R4 and R7 registers and outputting the result to R8) and has just fetched and decoded the instruction. Given the format above, determine the micro instructions necessary to carry out the **execute** and **writeback** phases of `ADD R4, R7, R8`.

Fill in the microinstructions on the empty lines first and then fill in the table:

1. R4out, X$_{IN}$
2. R7out, ADD, Z$_{IN}$
3. Zout, R8$_{IN}$

RF$_{\text{read}}$	**RF$_{\text{write}}$**	**RF$_{\text{addr-out}}$**	**RF$_{\text{addr-in}}$**	**F$_{\text{ALU}}$**	**X$_{\text{in}}$**	**MUX$_{\text{select(const/x)}}$**	**Z$_{\text{in}}$**	**Z$_{\text{out}}$**	**END**
1	0	0100	0000	00000	1	0	0	0	0
1	0	0111	0000	01101	0	1	1	0	0
0	1	0000	1000	00000	0	0	0	1	1

(b) Calculate the number of bits used for encoding in our simplified micro instruction set and explain your reasoning:

We have single bit encoded RFread, RFwrite, Xin, MUX, Zin, Zout and End which takes 7 bits and 2 4-bit vertically encoded RF address-in and out as well as 5-bit vertical FALU for a total of 20-bit encoded instruction set.