Solver: Shao Jie

Email Address: yews0012@e.ntu.edu.sg

1. (a)

(i)

M1 execution time = IC x CPI x clock period
= $2x10^9$ x 2 x (1/200 000 000)
= 20 s

M2 execution time = IC x CPI x clock period
= $2x10^9$ x 3 x (1/400 000 000)
= 15 s

(ii)

Speed up of M1 over M2 = M1 execution time/ M2 execution time
= 20/15
= 1.33

(b) Speed up of M1 = ((1-E)*20 + E*20/S) / 20
= ((1-E) + E/S)

(c) Speed up of M1 = ((1-U)*0 + U*20) / 20
= (0+ U*20) / 20
= U

(d)

```
        addi $t0, $zero, 200;
        addi $t1, $zero, 100;
loop    lw $t2, 0($a1);
        lw $t3, 0($a2);
        addi $t4, $ t2, $t3;
        sw $t4, 0($a0);
        addi $a1, $a1, 4;
        addi $a2, $a2, 4;
        addi $a0, $a0, 4;
        addi $t0, $ t0, -1;
        beq $t0,$t1, break;
        j loop;
break
```

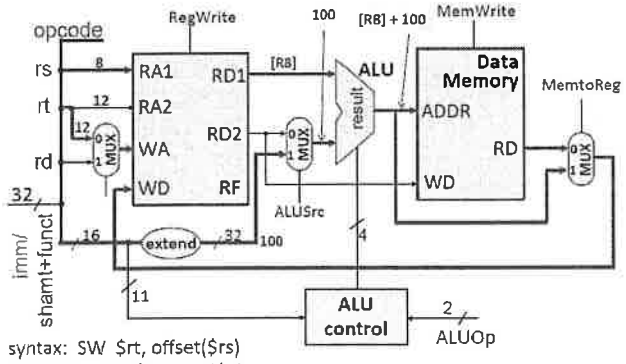
(e)

CISC	RISC
Emphasis on hardware	Emphasis on software
Includes multi-clock complex instructions	Single-clock, reduced instruction only
Memory-to-memory: "LOAD" and "STORE" incorporated in instructions	Register to register: "LOAD" and "STORE" are independent instructions
Small code sizes, high cycles per second	Low cycles per second, large code sizes
Transistors used for storing complex instructions	Spends more transistors on memory registers

2. (a)

(i)

datapath for ADD/ADDI/ SW/ Load Word instructions

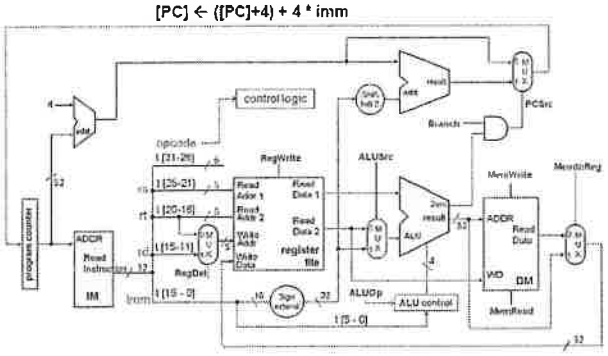


syntax: SW $rt, offset($rs)
example: SW $12, 100($8)
=> mem[[R8] + 100] ←[R12]

200ps + 1000ps+ 200ps + 1500ps + 2000ps +200ps =5100ps

datapath for ADD/ADDI/ SW/ LW and BEQ instructions

[PC] ← ([PC]+4) + 4 * imm



Red lines show the execution paths of BEQ instruction.

200ps + 1000ps + 200ps + 1500 =2900ps

(ii)

R-type: $1/2.9^{-9}$= 344.82 MHz

I-type: $1/5.1^{-9}$= 196.07 MHz

- maximum operating frequency $f_{max} \propto [V - V_{th}]^2/V$,
  - V_{th} is called the threshold voltage, the gate-source voltage at which the transistor just starts conducting. (Note that MOS transistor has three terminals: Gate, Source, and Drain)
  - if V_{th} is small compared to V then the maximum usable frequency $f_{max} \propto V$

(iii)

The voltage is reduced from 2volt to 1volt. Since the maximum operating frequency is directly proportionate to (V^2/V) which is V. Then (($V/2)^2$/ (V/2)) = (V^2/4) / (V/2) = V/2. Thus, maximum operating frequency is also reduced by factor of 2.

R-type: ($1/2.9^{-9}$)/2 = 172.41 MHz

(b)

(i) Full Data forwarding

1	F	D	E	M	W									
2		F	D	E	M	W								
3			F	S	D	E	M	W						
4					F	D	E	M	W					
5						F	D	E	M					
6							F	D	E	M	W			
7								F	D	E	M	W		
8									F	D	E	M	W	
9										F	D	E	M	W

(ii)

1	F	D	E	M	W															
2		F	D	E	M	W														
3			F	S	S	D	E	M	W											
4						F	S	S	D	E	M	W								
5									F	S	S	D	E	M						
6												F	D	E	M	W				
7													F	D	E	M	W			
8														F	D	E	M	W		
9															F	S	S	D	M	W

Without data forwarding

$s0 = 0x00000000, $s1 = 0x00001000, $s2 = 0x00000100

Signed Hex to decimal: 0xFFFC = -4 → -4

Total number of iterations: 100/4 = 25

Steady state CPI = (No of instructions + no of stalls)/ No of instructions

Steady state CPI (without data forwarding) = (9 + 8)/ 9 = **17/9**

Steady state CPI (with data forwarding) = (9 + 1)/ 9 = **10/9**

3. (a)
(i)

```
        addi $s1, $zero, 100
Loop:   lw $t1, 0($t0)
        addi $t1, $t1, 8
        sw $t1, 0($t0)
        addi $t0, $t0, 4
        addi $s1, $s1,–1
        bne $s1, $zero, Loop
```

addi $t0, $t0, 4	lw $t1, 0($t0)
addi $s1, $s1,–1	nop
nop	nop
addi $t1, $t1, 8	nop
nop	nop
nop	nop
bne $s1, $zero, Loop	sw $t1, 0($t0)

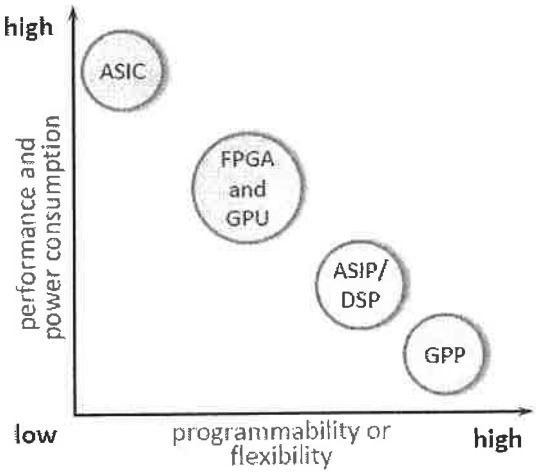
2 way

CPI = 7/6 (1st instruction is not included because it is only ran once, thus it is not included in the steady state CPI)

(b)

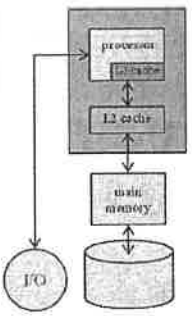
Comparison of Processor Technologies

Performance and power consumption vs flexibility



General Purpose Processor (GPP)

- Hardware features
  - the program to be run and necessary data could be made available at the main memory
  - general data-path: consists of a general ALU and usually a large register file
  - multiple levels of cache for reducing memory latency
- Maximum flexibility
  - programmable: supports several high-level languages
  - can be used for any general application
- Other key features
  - short time-to-market
  - low NRE cost
  - high-power consumption

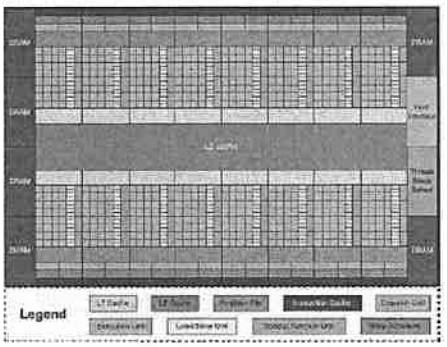


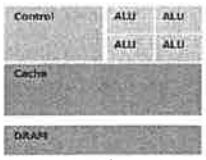
Application-Specific Instruction-Set Processor (ASIP)

- Application-specific instruction-set processor (ASIP): a microprocessor tailored to benefit a specific application or a domain of applications
  - Ex: signal processing, image processing, video processing, and digital communication, etc.
- Instruction set of ASIP is customized for the type of computation involved in the specific application
- Requirement on flexibility should be just sufficient instead of unlimited like general purpose processor
- To achieve highest performance with minimum of power consumption, silicon cost and design cost

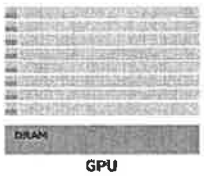
Graphic Processing Unit (GPU)

- GPU is a processor –specialized for graphics.
- GPU is flexible to implement any algorithm instead of being only for graphics to have more utilization efficiency and high performance: GPU computing
- Heterogeneous execution model(CPU is the *host*, GPU is the *device)*
- Programming model is "Single Instruction Multiple Thread" (SIMT)





CPU



GPU

ftp://download.nvidia.com/developer/cuda/seminar/TDCI_Arch.pdf

Graphic Processing Unit (GPU)

- Motivation of GPU computing
  - parallel computing by massively *data parallel stream processing* [1]
  - using *thousands of threads.*
  - implemented in *hundreds of cores*
  - cost-effective: cheap
  - using already available hardware
- GPUs involve high latency and provide high throughput processing
- CPU performs low latency low throughput computation
- Embedded systems are usually real-time

For reporting of errors and errata, please visit pypdiscuss.appspot.com
Thank you and all the best for your exams! ☺