Solver: CHEN BENJAMIN, HOANG VIET, WILLIS TEE TEO KIAN, YONG SHAN JIE

(Not going to lie, this paper is very difficult, and some answers cannot even be found in lecture material. Some answers are extracted from tutorial, you may refer to tutorials for similar examples)

1)

a)

	Embedded system	**PC, laptops**
Memory	Memory is often limited; hence programmer needs to be mindful of memory allocation	Memory is generally not a concern as external memory can suffice the needs
Power	Power is usually limited due to supply by battery; hence power consumption is a major issue	Power is provided through high capacity battery or directly from power source
Function	Usually have to input/output from different peripherals (sensors, actuators etc) and process input within specified time constraints (e.g DSP system), thus needing special interfaces such as dedicated DAC/ADC.	Additional resources such as memory, I/O, peripherals, etc. must be interfaced with the microprocessor

b)

i)

0x2000 in binary is: **0010** 000000000000

0xA000 in binary is: **1010** 000000000000

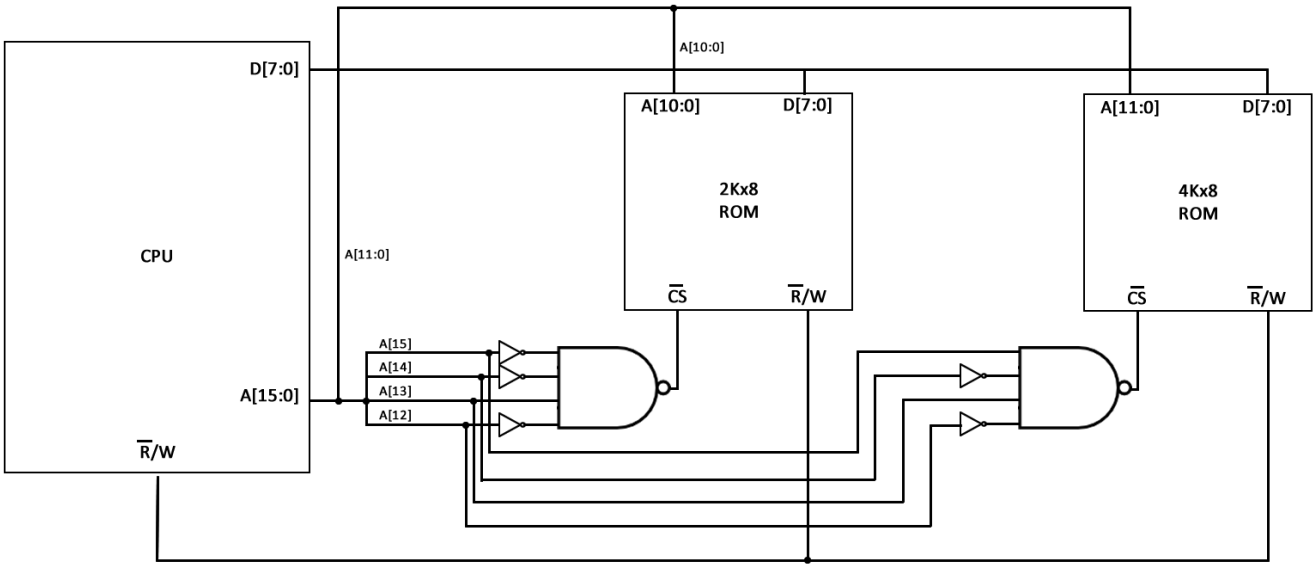

Figure 1. Partial Address Decoding Circuit

ii) Note that there are 2 instances of 2Kx8 ROM because partial addressing is used (ambiguity for A[11] is not resolved), hence both 0x2000 and 0x2800 can access the same ROM.

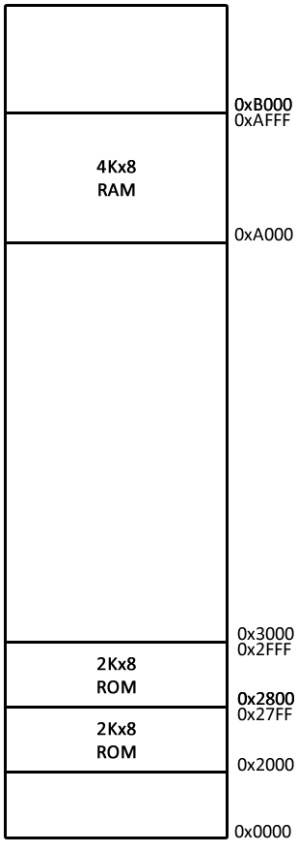

Figure 2. Memory Space

iii) Line A[11] added to the NAND gate to fully address 2Kx8 ROM.

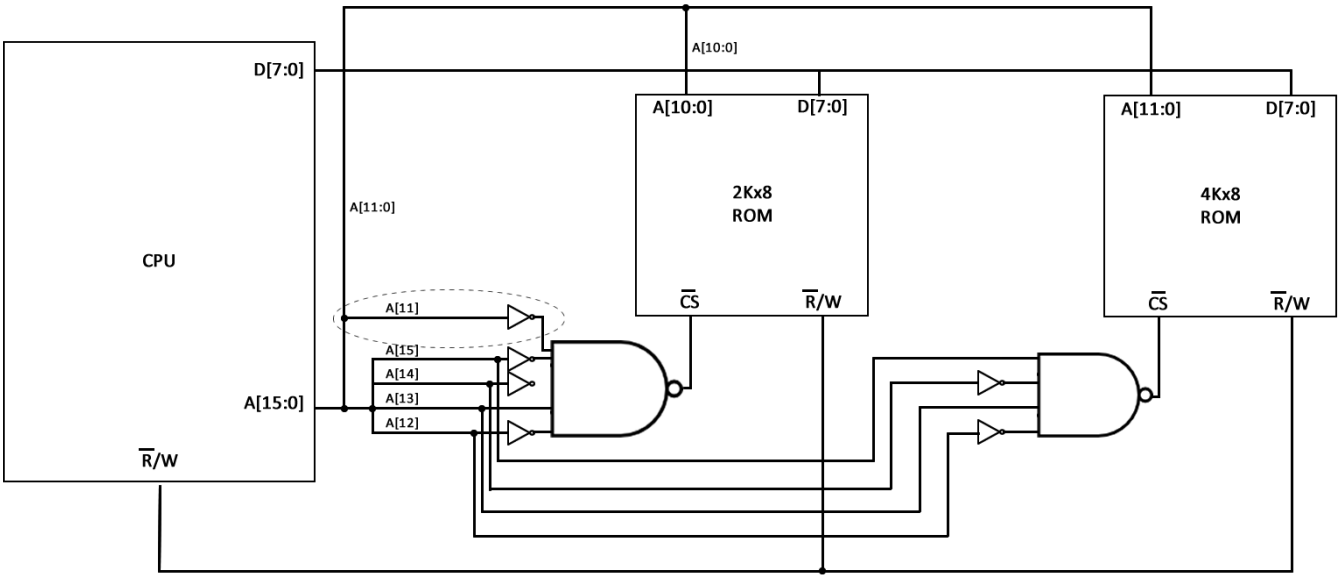

Figure 3. Full Address Decoding Circuit

c) Line 1: **Data**. It is a declared global variable
Line 2: **bss**. It is an undeclared global variable
Line 3: **Data**. It is a declared global variable
Line 4: **bss**. It is an undeclared static variable
Line 5: **Stack**. It is a local variable.
Line 6: **bss**. Declaring a variable does not shift data from bss to Data.
(https://stackoverflow.com/questions/36466317/is-global-variables-travel-from-bss-to-data-segment)

2)

a)

i) Full scale = 1V - 0V = 1V

Quantization size = $\frac{1V}{2^4}$ = 0.0625V

Depends on the configuration of the comparator you may get a different intermediate working. Here, we assume that V_{DAC} is connected to the Negative Terminal, and analog input is connected to the Positive Terminal of the comparator.

Bit to be compared is first '1', if Vin is greater than SAR value, the respective bit will be 1, else it will be 0.

Iteration	**SAR value**	**DAC value**	**Comparator output (Is 0.23V > DAC Value?)**
1	**1**000	0.5V	0
2	0**1**00	0.25V	0
3	00**1**0	0.125V	1
4	001**1**	0.1875V	1

Final output of SAR is 0011 corresponding to **0.1875V**

ii)

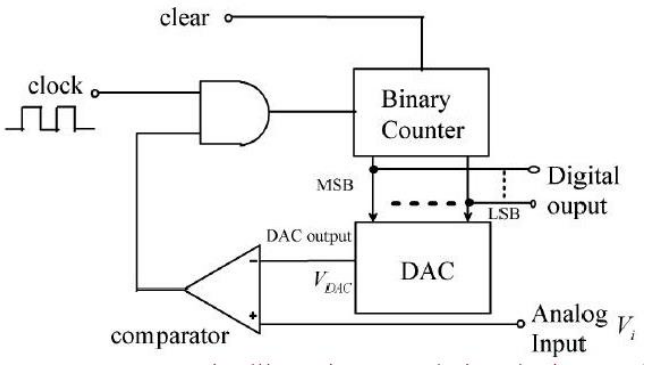

Figure 4. SAR Decoder Circuit, Adapted from Chapter on ADC

b)

i)

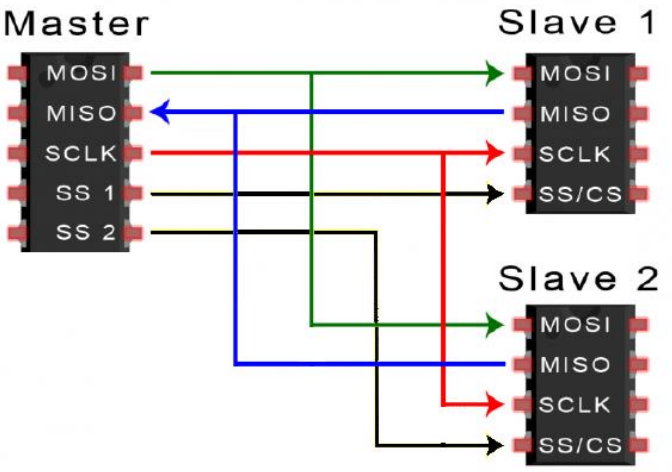

Figure 5. SPI Communication, Adapted from Chapter on Serial Interface

ii) Master will always initiate the communication, and the one responsible for generating the clock (through SCLK). In each clock cycle, 1 bit of data is transmitted. Master first choose which slaves it wishes to communicate with by pulling the respective SS line LOW. Master will then transmit data to slave through the MOSI line and receive data from slave through the MISO line. Data transmission is done MSB first.

c)

	Common Anode	**Common Cathode**
HW consideration	Requires Current Limiting Resistor	
	Ensure that microprocessor can sink all the current from all segments.	Microprocessor must be able to source enough current to illuminate all segments.
SW consideration	Refresh rate must be sufficiently fast for multiple 7-seg-display that are multiplexed, so that all the displays appear to be on to human eye.	
	Writes ‘0’ to illuminate segment	Write ‘1’ to illuminate the segment

3)

a)

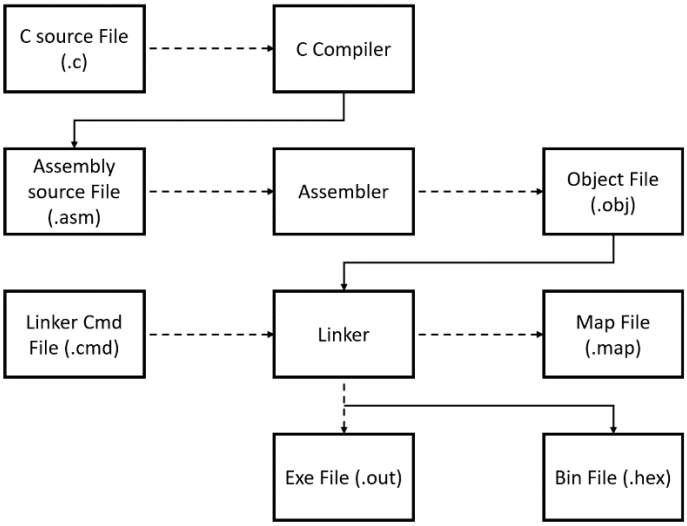

Figure 6. Microprocessor Software Development Toolchain Adapted from uP Toolchain Chapter.

We will walk through the diagram following the arrows, starting from C source file

File/Hardware	**Functions**
.c	High level language file written by programmer
Compiler	Translate the .c file into assembly language (.asm). Assign code/data to various default and user-defined sections.
.asm	File output of assembler. Contains Assembly instructions
Assembler	Convert .asm file into binary file (.obj) that can be recognized by a specific type of processor.
.obj	Contains machine code with symbolic address references.
.cmd	A file reference by linker when assigning absolute addresses to different software sections. It contains configurations of the target processor's physical memory and software sections to physical memory mapping information as well as other miscellaneous linker parameters.
Linker	Allocate code/data to specific location in the memory and generates an executable file with fully resolved address references
.map	Shows the absolute address information of the memory, sections and symbols defined in the program
.out	The result of several object files linked together. It contains additional metadata and runtime features, hence normally contain significant additional machine code than the source code.
.hex	Contains machine codes of the program, together with associated information such as the section, address of each section. It is coded in ASCII text form for readability and is used typically as an input file to programmer for programming into processor or standalone system

	memory during production. As such, the machine code are typically partitioned into smaller chunks with CRC code to verify its integrity during programming

b) To ensure the voltage supply complies to the specification of the processor. This is to prevent random & potentially disastrous behaviour.

c)

i) **Tail-chaining**: Every interrupt handling begins with state-saving and ends with state restoration. During the end of an interrupt, the tail-chaining mechanism can check if there is another pending interrupt. If there exist pending interrupts, the state restoration and saving between two interrupts is skipped. States will only be restored when all interrupt is serviced. This speed up interrupt servicing.

Late Arrival: During State Saving of an interrupt, if there is another interrupt of a higher priority that is being triggered, the State Saving process will carry on instead of restarting. However, the vector of the higher priority interrupt is fetched, and the first interrupt to be serviced is the one of higher priority. After the higher priority interrupt is completed, normal tail-chaining rule applies.

ii) Assumptions: Processor has Tail-chaining & Late-Arrival mechanism, and 6 cycles is incurred for vector fetch between interrupts. (These 6 cycles incurred is a behaviour of Cortex-M4. Refer: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0439b/ch03s09s01.html)

t	Event
2	IRQ2 Triggered, Context Save Begins
10	IRQ1 Triggered, Context Save is still being carried out
14	Context Save Complete, Since IRQ1 Priority Higher, IRQ1 is executed first
44	IRQ1 Complete. Fetch vector of IRQ2
50	Begin execution of IRQ2
70	Completion of IRQ2. Restore States.
82	State Restored.

Total cycles required from first context save to context restore = 82-2 = **80 Cycles**

d) Clock Source = 32kHz

Timer Clock (After Divisor) = $\frac{32kHz}{32} = 1kHz \rightarrow Period = 1ms$ (Each count takes 1ms)

Total Period = 40ms + 60ms = 100ms. Since Timer is in Up-Down mode, it will spend half the time counting to CCR0, and the other half the time counting back down to 0.

$$CCR0 = \frac{100ms}{1ms} \div 2 = 50$$

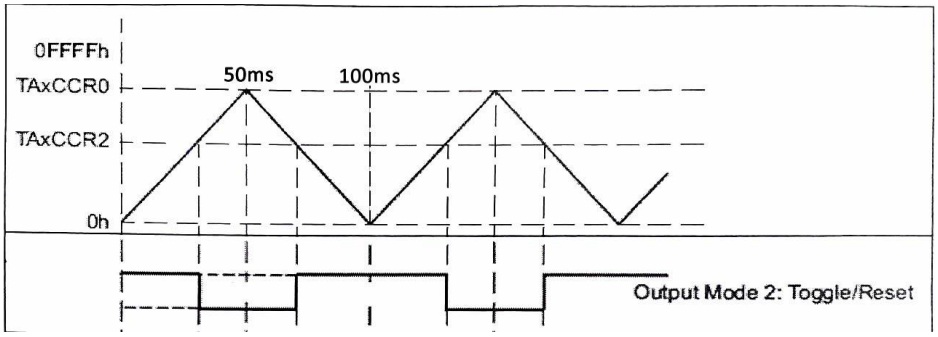

Figure 7. Timing Diagram, time when CCR0 is reached, and time when counter reach 0

Since output needs to be low for 40ms,

$$CCR2 = \left(50 - \frac{40ms}{2}\right) \div 1ms = 30$$

TA1CCR0 = **50**, TA1CCR2 = **30**.

4)

a)

i)

Peripherals	**Interface/Middleware**	**Explanation**
Audio Speaker	Codec interface/ Audio Processor (ADC/DAC)	To convert analog audio signal (from microphone) as well as output (to speaker)
LCD	LCD interface / Display Controller (Example: HD44780 Controller)	Takes input data as instructions and perform an action based on the command.
NAND flash	Address Decoding Circuits	A circuitry to map the processor address space to physical memory components by using specific bits of the processor's address to select the components.

ii) Zero-wait state means NAND flash can keep up with the speed of the microprocessor. In brief, when the microprocessor issues an instruction for any NAND operations, it does not have to wait any additional cycle for the NAND to response - the NAND operates at the same frequency as the processor.

iii) Error Correction Code is needed due to the high density of cells in one chip. Random corruption may also occur over time due to wear and tear from program/erasure operations.

b)

i) Processor = 120MIPS = 120MHz (assuming 1 instruction/cycle)
Power Consumption of Processor = 3mW/MHz * 120MHz = 360mW
Total Power Consumption = 360 + 300 = 660mW
Battery Capacity (mAh) = 660mW / 3.7V * 12 hrs = 2140mAh

ii) Self - Refresh Mode. This is when the clock is killed and (assumed) deep sleep mode is enabled. This allows lower power consumption (as there is no clock) and full data retention can be optimized via adjusting optimal refresh frequency.

iii) Floating Pin issue: Under normal operation, the input would be pulled to a high or low logic state, which would enable one of the transistors and disable the other. But when the input is left floating, i.e. not tied to GND or VCC, then it could take on any voltages and there is a possibility that it would have a voltage mid-way between VCC and GND. This would cause both transistors to be ON and a high current would flow from VCC to GND directly. Recommendation is to configure the pin to output mode if it's a bi-directional pin type or tie the input pin to a known state (Either Vcc or Ground).

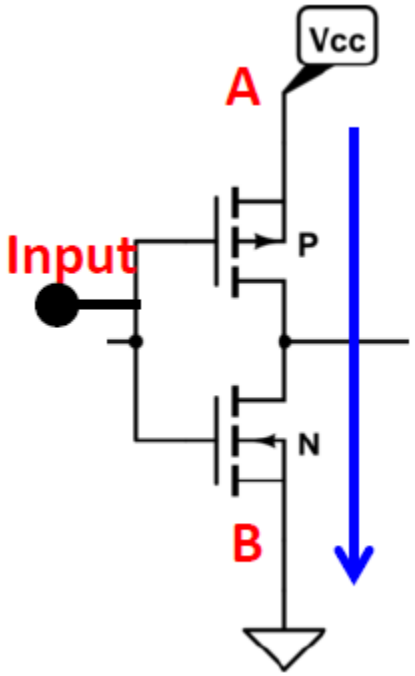

Figure 8. Vcc Potentially shorted to Ground. Adapted from Design Consideration Chapter

c) ARM's big.LITTLE configurations have two main value propositions:

Optimal resource deployment	• ARM big.LITTLE™ technology is a heterogeneous processing architecture which uses two types of processor: "LITTLE" processors are designed for maximum power efficiency and "big" processors are designed to provide maximum compute performance. • Processor vendor is allowed to mix and match different number of big and LITTLE cores to suit their targeted application's loading. • When the application loading is light, the LITTLE processor will be used to save power. When the loading when beyond that of the LITTLE processor, the big processor will be switched in to carry the heavier load.
Dynamic and efficient adaptation to application load	• One of the main issues with heterogeneous processing, or in fact, that of multi-processing are the data paths between processing cores. It is usually slower than the processor core speed so tends to be the bottleneck. Especially if one of the main features is to be able to dynamically move program around the system. • ARM build a special interconnect known as the" Cache Coherent Interconnect" (CoreLink GIC-400) enable seamless data transfer between clusters, particularly between cache of each processors, without having to go through relatively slow external memory • Employment of Global Task Scheduling allows full flexibility to allocate a particular task to any available processor core.

--End of Answers--