August 23rd (Mon).

CMPE240 Section 1.

CMPE240

Harry Li.
E-mail: hual@sjsu.edu
Text message (650) 400-1116

Office Hours: M.W. 3:40-4:40 pm.

Advanced Microprocessor Systems = Prototype System with A CPU module

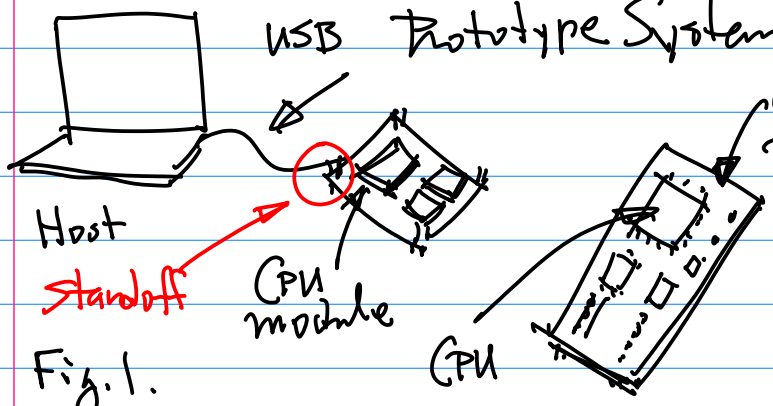


Fig. 1.

GPU (Graphics Processing Unit), Array of Processors, Machine Learning, AI. Autonomous Systems. Nvda Jetson TX2.

Textbooks, References

1° NXP LPC1769 CPU Datasheet 80 pages Homework: Down Load pdf. Before Next Monday. Aug. 30th.

2. LPC1769 schematics of the CPU module

3. Nvda Jetson NANO Datasheet on TX2 (6 CPU + 256 GPU) 400 pages. (optional) 5% Bonus.

4. RISC-V. Open Source Architecture, A super set of ARM. FPGA, Verilog. SoC. +RTOS. (optional) +5%

A proposal (One Paragraph) By Sept. 1st (Wed). submit to my email;

Note: Buy LPC1769 CPU module.

digi-key.com, mouser.com, etc.

Grading Policy & Projects

2 projects (phase I & II)

2D Graphics Processing Engine → 3D Graphics Processing Engine { a, b, c

2D Graphics Processing Engine → 3D Graphics Processing Engine { 3D shadow. 10% / Diffuse Reflection 10%

10% = 30pts, 30%.

midterm: 30%, Final 40% (Comprehensive)

Option 1. (5%+) NVDA NANO
- a. Likely Devices Drivers, O.S. C/C++, Python.
- b. I/O Interface: "EdgeAI" GPIO, SPI.

Option 2. (5%+) RISC-V Target SoC, FPGA Board,

Proposal (one paragraph), Submission By Sept 1st (Wed) via E-mail.

Policy ON Project Submission.

1°. Form 3-4 person Team.

2° No Source Code/Design material Can be Copied; All Course material has to be completed individually;

3° Late Project, 10% per week;

August 25 (Wed)

Today's Topics:

1° Bill of material

Reference: github/hwalili/Cmpe240/2018F

The B.O.M.

1. CPU module NXP LPC1769
   - 3rd Party (DigitalArt), module
   - To Distributors Digikey.com, Mouser.com etc. Expecting Delays. Lead Time over 8 weeks
   - Alternative { Re-use the previously used module / Team (4 person)

Each person will need to have his/her Board;

Option 1: NANO. ≈ 4400+ pages Datasheet "firmware"

≥ Jetpack 4.3 or Higher (O.S. + Libs. + Packages)

Tool for Flashing the CPU module

c. Coding in Both user & Kernel Spaces. → O.S. Distr. Toolchain, Device Driver Debugging & Development;

Option 2. RISC-V. verilog, FPGA.

2. Power Regulator IC series 7812, 7805 ... 1117

78xx

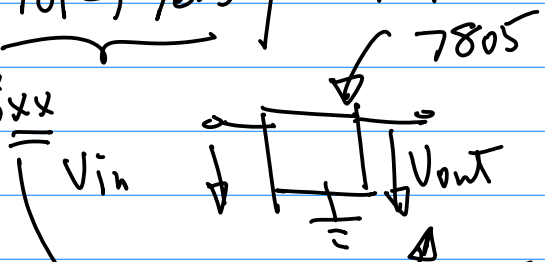


Fig.1.

"05": 5.0 VDC, "12": 12VDC

Fig.

$V_{in} \geq V_{out} + 1.5\,VDC$ ... (1)

DC Voltage Source

a. About 7805 1000 mW.

b. 7.5VDC OR 9.VDC

Fig 2

@ 1000mW + 500 mW = 1500 mW

Current Rating.

c. Why Do we use it?

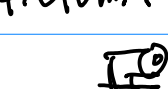
→ Deploy the System.

3. "Glue" Components < Resistors a. Caps 4.7 mF b.

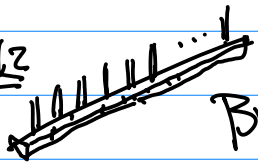
c. LEDs. (Red, Green) for Debugging purpose, for PWR. (GPIO), I_{LED} = 4~10mA

d. Connectors.

d1. J1 for PWR Input



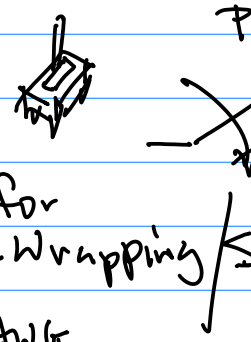
d2.



to mount CPU module.

e. Switch. S/W1: to toggle PWR.

S/W2



f. Wire for Wire Wrapping / Soldering 28-30 AWG

4. Color LCD Display module

a. SPI (Serial Peripheral Interface)

b. Software Graphics (Driver) C/C++ Lib. to Activate/Initialize LCD.

MCUXpresso (I.D.E.)

S.T. Lib.

5. "Other" thing.

RJ-45 Connector Right Angle

CMPE240

August 30 (Mon)

Today's Topics.

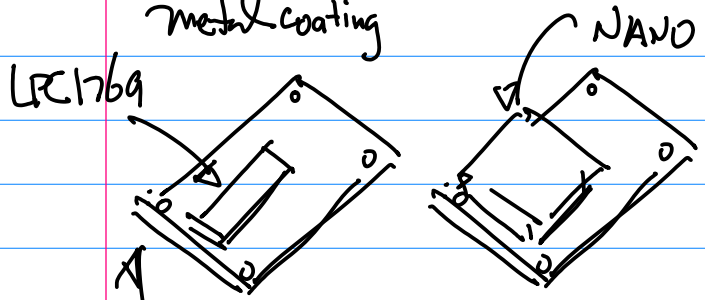
1° Introduction

Example: To Build Prototype Board

1° Get A Wire Wrapping Board

a Dimension 4"x3"

b. Refab Through-Holes, with metal coating



c. 4 Mounting Holes Fig 1.

d. 4 Stand-offs (Legs)

For Both LPC1769 And NAND.

2° Build "Hello, the world"

a. IDE MCU Xpresso

www.nxp.com

Homework: Download/Install MCU Xpresso

Aug 30.

C/C++ Code print "Hello..." for Target platform.

b. Flash the program to make a Simple firmware Load to the target Board, to execute the program;

c. GPIO output { "1" Turn on LED; "0" Turn off LED

GPIO Input { Read "1", when SW → V_{cc}; Read "0", when SW → GND

3° Design in Hardware

a. Power distribution to the System

{ CPU PWR; Subsystem(s) PWR

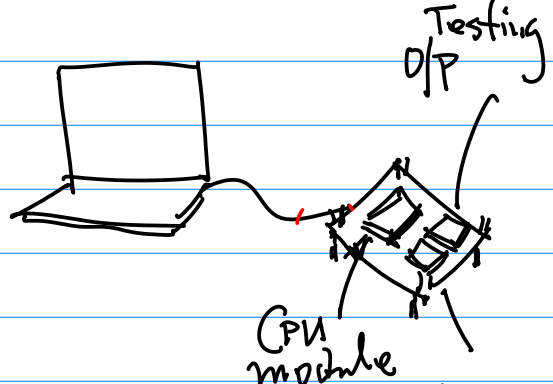


Fig 2

USB Cable: Laptop to CPU module

Option 1. PWR from the target platform

SCH provides information for the Target Board.

Common Practice for the target Board is to provide Vout (3.3VDC or 5.0VDC)

CMOS TTL

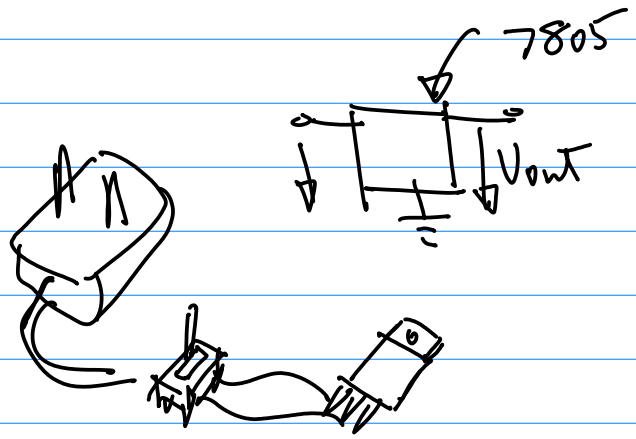


Fig.4

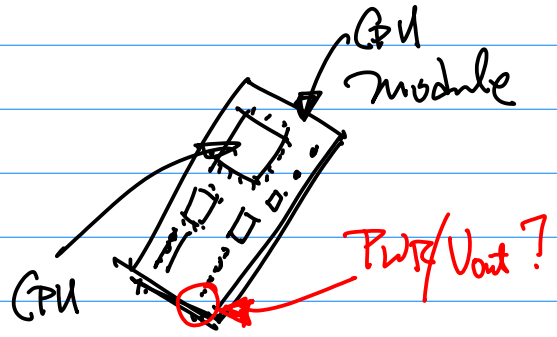


Fig3

IC PWR Regulator Such as 7805

Note: for NAND

First, Becomes a developer;

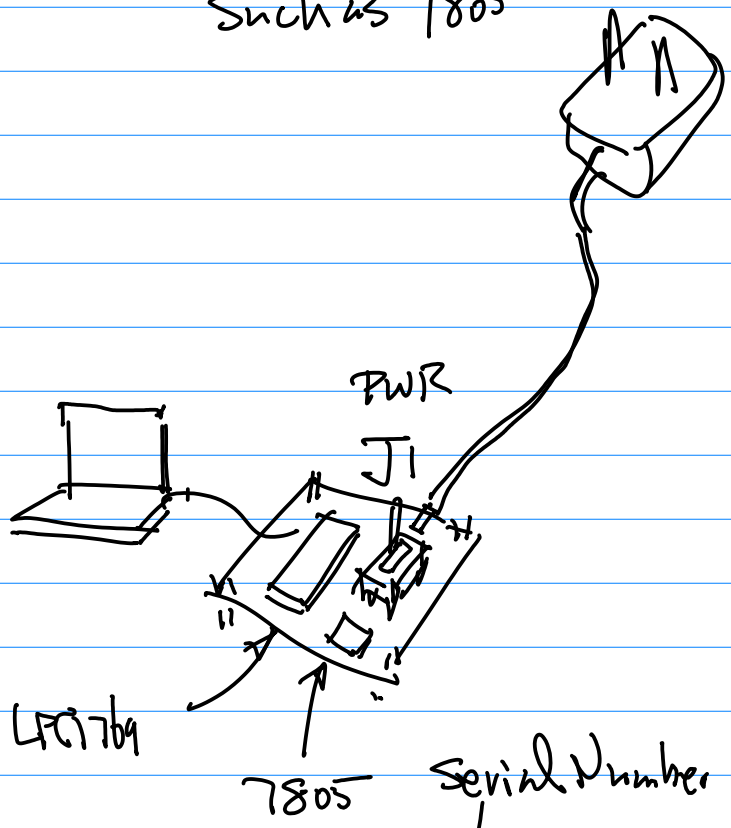
Then, DownLoad/Install

JetPack 4.3

(O.S. + Lib + Deep learning)

↓

Ubuntu 18.04



Option 2. External PWR Supply

Sept. 8 (W)

Topics: 1° "Hello, the world" program

- Hardware Implementation
  - NXP MCU Xpresso.
    - a. Installation of MCU Xpresso.
    - b. github/hualili/CMPE240/2018F
      LPC1769 Patch, Import this patch to your Xpresso.
  - Prototype Board Build up
    - a. Wire wrapping Board (LTC/NAND/Pic)
    - b. Stand-offs.
  - External Power CKT (Red LED should be included)
  - GPP Testing CKT
    - Implementation/Design of the CKT.
    - Architecture Aspects, CPU Architecture, M. Map.

https://github.com/hualili/CMPE240-Adv-Microprocessors/blob/master/1769%20patch.zip

Note: Wire Wrapping Board with "Stand-off's (Legs)

Homework: Next Show-and-tell Wire Wrapping Board;

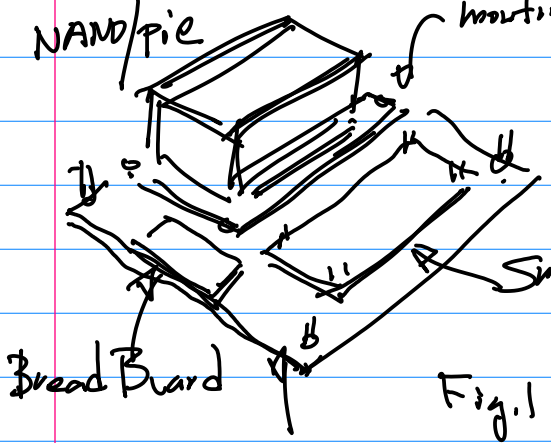


Fig. 1

Wire Wrapping Board Carrier "Board"

"TAP Plastic"

On the Board: a Stand-offs.

b Connector(s) for External PWR

PWR IC (7805), with Red LED

CPU Architecture:

1. 32-bit Architecture

CPU Architecture

a. ALU 32 bit Arithmetic/Logic Unit.

b. Register "File",

A Bank of Registers. 32 bits

GPRs

General Purpos Registers

Those Registers that can participate Any meaningful

Arithmetic/Logic Operations.
Special-purpose Registers.
SPRs 32 bit

To Define/Determine the Behavior of Peripheral Controllers.

Naming Convension: 6 letters

Common Design for SPRs:

1° Control Register(s) per Each Peripheral Controller

CON ⟸ Root (3 Letters)

2° Data Register, DAT

3° Pull-up/Down (Electric Characteristics)

C. Data Bus, "Bi-Directional" 32 bits
Information Flowing Both Directions.
Address, "Uni-directional" from CPU to the Outside. 32 bits

Notation: 32 bit Register

$GPR_X[31:0]$

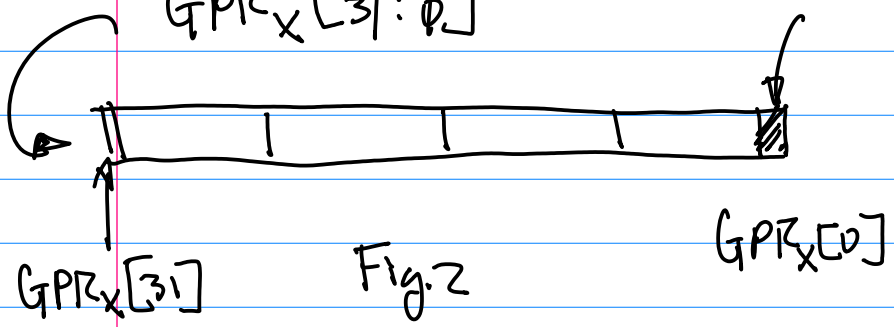


Fig. 2

For Address Bus, $Addr[31:0] =$

$a_{31}\, a_{30} \cdots a_1\, a_0$

Note: "Little Endian"
LSB is a 0,

2. "Byte Addressable" machine is a machine whose Smallest memory cell with an unique address is a single Byte.

Total memory:

$$2^{32} = 2^2 \cdot 2^{10} \cdot 2^{10} \cdot 2^{10} \quad \cdots (1)$$

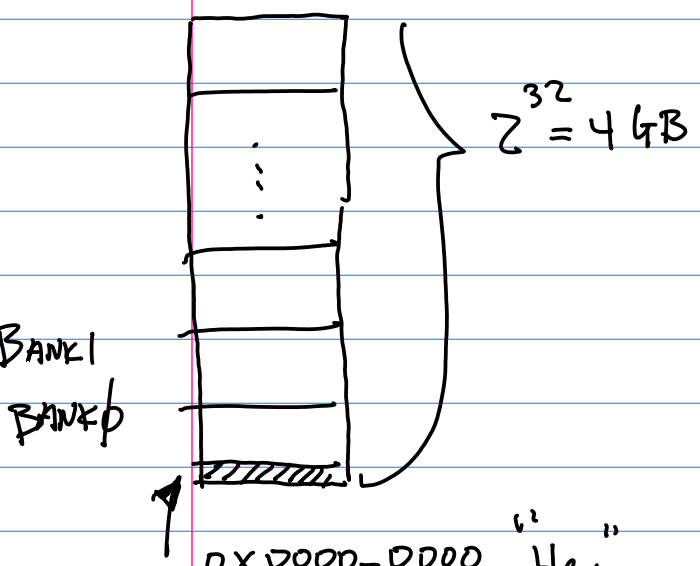
$2^{10} = 1K$, $\cdots (2)$

$2^{20} = 2^{10} \cdot 2^{10} = 1M$ $\cdots (3)$

$2^{30} = 1M \cdot 1K = 1\,Gigm$ $\cdots (4)$

$2^{32} = 2^2 \cdot 2^{30} = 4 \cdot GB$

3. Memory Map.



0X0000-0000 "Hex"

32 bits for the Address
8 bits for this memory

a. PWR-up Address:
CPU will fetch the 1st Executable from this memory Location.
→ 0X0000-0000 for ARM

Note: for x86, the PWR-up Address: 0XFFFF-FFF0

b. BANKS. $2^{32}/8 = 2^{32}/2^3$

$= 2^{29} = 2^9 \cdot 2^{20} = 512MB$

How many Bits Do we need to uniquely define Each Bank?
3 bits → $a_{31} a_{30} a_{29}$

Define Starting Addr. of Each Bank:

a_{31}	a_{30}	a_{29}	a_{28}
0	0	0	
0	0	1	
0	1	0	
⋮			
1	1	1	

BANK0
BANK1
BANK2
⋮
BANK7

Fig 3.

Write the Address for Each Bank. "Starting" (32 bit)

For BANK0 : 0X0000-0000
BANK1 : 0X2000-0000
" 2 : 0X4000-0000

Exmple: CPU Datasheet pp. 13.
GPIO 0X2009-C000

a. Collection of SPRs are mapped to here, e.g. Addr. for SPRs are mapped to here

b. Which memory Bank holds this GPIO? BANK1 whose starting Address is 0X2009-C000