Coverage for midterm:

1. Everything up to and including Thursday's lecture
2. All 4 labs, all lectures
3. Expect full understanding lab processor @ digital logic level (Verilog)
4. Expect know all 8 instructions for Lab 2/Enhanced processor
5. Expect know ARM instructions used in Lab 3&4

Review:

1. Information representation (binary, hexadecimal, 2's complement, ASCII)
   A. ASCII: numerical codes for letters, numbers, symbols that come from a keyboard input.
      e.g: F = 0x46 E = 0x45 D = 0x44
      codes are sequential for the uppercase & lowercase alphabet

   Just as we can, in assembly program "declare" data:
      data .word 0x12FA

   If data is at address 20,
      20 0x00 00 12 FA //32 bit word, 4 bytes (23 21 22 20)
      the least significant byte is at the lower address -> "Little Endian"
      In the opposite of "Big Endian"

   You can also declare ASCII code data easily in assembly,
      24 lets: .ASCII "FEED"
   word @ 24 = 0x44 45 54 56 (37 35 36 34)

   say R0 = 21: STR r4, [r0]
   if "store" at an unaligned word boundary (that is multiples of 4) -> processor error (exception)
   but STRB r4, [r0] -> can access any addressable byte
   also if load from unaligned address, also get error.

LDR R0,[R1]: loads the word at the address given by R1
LDRB R0,[R1]: loads the byte @ address given by R1 & sets rest of R0 to 0
LDRSB R0,[R1]: load byte & signed-extend: copy bit 7 (sign bit) of byte into bit 8-31

Lab 2/Enhanced processors vs. ARM processor

Registers: r0-r6 general
r7 program counter
N
Z
C condition code bits
16 bit registers
word addressable

r0-r12 general purpose
r13 stack pointer
r14 link register
r15 program counter
CPSR condition code flag (N, Z, C, V)
byte addressable
"little endian"

load/store architectures: must load memory value into a register before can operate on it.