159233 Tutorial 1

Section A

1. How many possible three input, one output binary logic functions are there?

   a) 256 b) 64 c) 8 d) 16

2. The following truth table is for which logic gate?

x	y	z
0	0	0
0	1	1
1	0	1
1	1	1

   a) AND b) OR c) NOT d) XOR

3. Which of the following is NOT true?

   a) An eight bit adder can be made from two four bit adders
   b) An adder is a sequential logic device
   c) A full adder has three inputs
   d) A four bit adder can be made from four full adders

4. What is the truth table for the borrow output from a 3 bit subtractor (performing x-y-z)

   a)

x	y	z	b
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

   b)

x	y	z	b
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

   c)

x	y	z	b
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

   d)

x	y	z	b
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

5. What is the name for a logic device that can choose one of a number of inputs as its output.

   a) Decoder b) Multiplexor c) Demultiplexor d) XOR gate

Section B

6. For the following truth table:

a	b	c	d
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

   Give a boolean expression for d in terms of a,b and c. Draw a logic diagram for this function.

7. Redraw the logic diagram for question 6 using only NAND and NOT gates.

8. Design a voting circuit with four inputs and one output, the output should only be one if three or more of the inputs are one.

   Draw the truth table, give a boolean expression and draw the logic diagram for this function.

9. Design a 4 bit subtractor to subtract 2x4 bit signed (twos complement) numbers and generate a 5 bit signed number. (hint: a-b=a+(-b),-b=NOT b+1)

159233 Tutorial 2

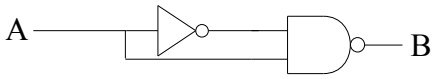
Section A

1. What is the following boolean expression.

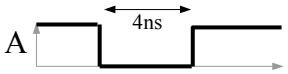
   $C=(A+B).\overline{(AB)}$

   a) NOR b) NAND c) XOR d) XNOR

2. Given the following circuit, a negative pulse of 4 ns and a gate delay of 1ns

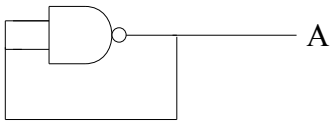


   If A changes as follows,



   Draw B on this diagram:

3. What is the output of this circuit?



   a) A is always 0 b) A could be 1 or 0
   c) A changes quickly between 1 and 0 d) A is always 1

4. For an RS flip flop, if Q=1,S=1 and R=0 what will happen if R becomes 1

   a) Q will stay 0 b) Q will become 1
   c) Q will oscillate between 1 and 0 d) Q will become 1 or 0 at random

5. When does the output from an negative edge triggered flip flop change

   a) When the clock is 0 b) When the clock is 1
   c) When the clock goes from 0 to 1 d) When the clock goes from 1 to 0

Section B

6. Draw a diagram to show how a D-type flip flop can be built using a JK flip flop and a NOT gate.

7. What is the purpose of the R/W input to a memory device?

8. Show how D-type flip flops may be used to build a four-bit register?

9. What is a tri-state device? In what situation is such a device commonly used?

10. Draw a circuit for a memory with a 2 bit address bus and a 2 bit data bus.

159233 Tutorial 3

Section A

1. A shift register is usually made from

   a) Edge Triggered JK flip flops b) Level Triggered JK flip flops
   c) Edge Triggered D flip flops b) Level Triggered D flip flops

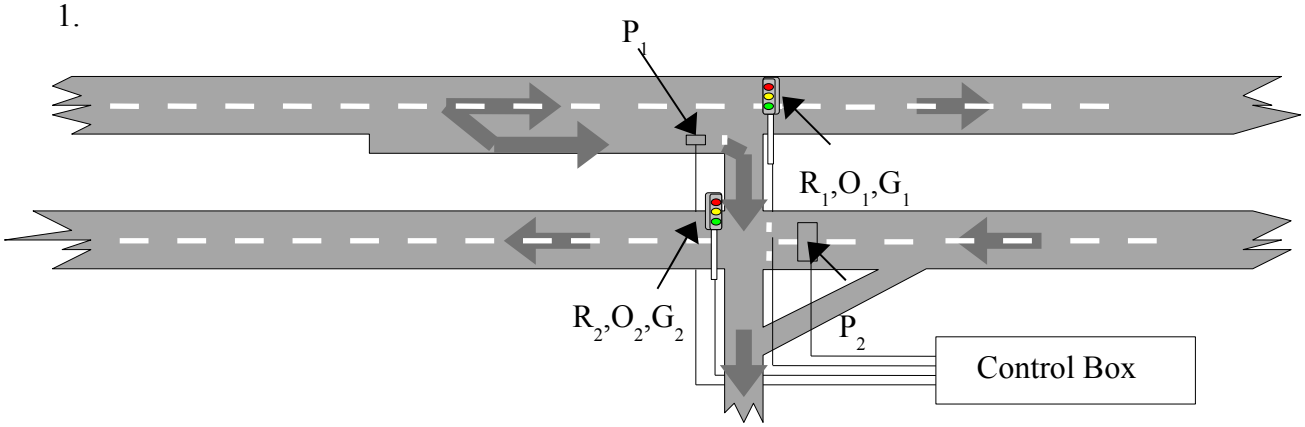
2. Draw a circuit for a 3 bit asynchronous counter

3. Draw a circuit for a 3 bit asynchronous down counter

4. Draw a circuit for a 3 bit synchronous counter with parallel load.

Section B

1.



A simple road junction is shown above. The main road is a highway with two lanes on each side. The junction is with a one way road and traffic lights are used to allow vehicles to cross the oncoming traffic.

The control box has two inputs P_1 and P_2 that are connected to sensors under the road. These have a value of 1 if there is a vehicle present above the sensor and 0 otherwise. The box also has outputs which go to two traffic lights (light 1 and light 2). Each traffic light contains three lamps, red, orange and green, thus the outputs are composed of six signals R_1, O_1, G_1 and R_2, O_2, G_2. The control box has a clock with a cycle time of 5 seconds.

If there are no vehicles present on P_1 then R_1, and G_2 will be on. When a vehicle is detected by P_1 then light 2 will turn orange. After five seconds, light 1 will turn green and light 2 will turn red. The lights will stay in this state while there is a vehicle on P_1 and there is no vehicle on P_2. When this is no longer the case, then light 1 will turn orange. After five seconds, light 1 will turn red, light 2 will turn green and traffic will flow along the highway as before.

(a) Draw an ASM chart for your controller.
(b) Give Boolean expressions for the states.
(c) Draw your multiplexor driven controller

2.
A child's toy is being designed to help increase co-ordination skills.
It has three buttons (red_button, green_button and blue_button),
and three lights (red_light, green_light and blue_light).

The toy's controller (which has a cycle time of 1 second) obeys these rules:

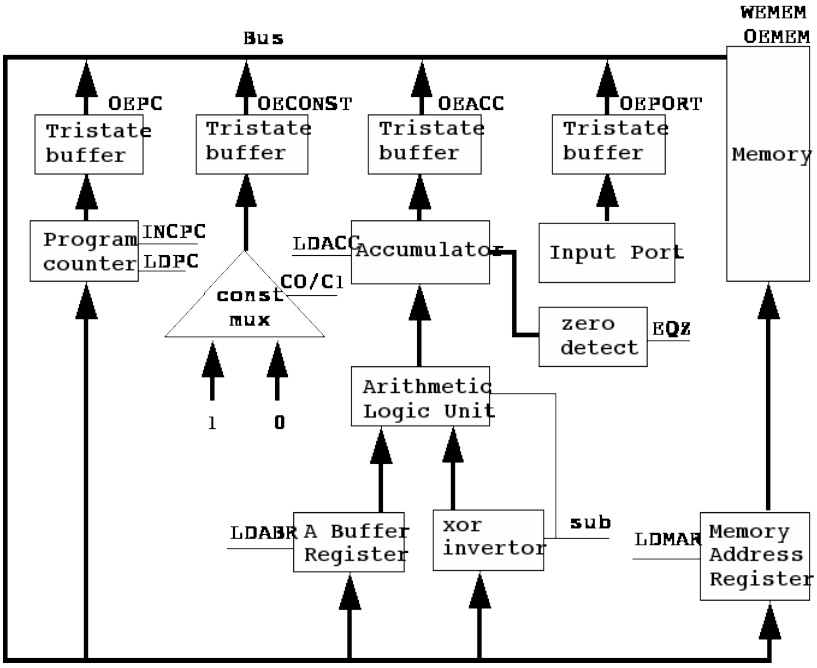
- Wait until the child presses one or more of the buttons.
- If the child presses one button, the corresponding coloured light is lit for 1 second.
- If the child presses two buttons together, the light with the same colour as the button that has not been pressed is lit for 1 second.
- If the child presses all three buttons together, all the lights are lit for 1 second.
- Go back to waiting.

For the controller of the above toy:

(a) Draw an ASM chart
(b) Give Boolean expressions for the states.
(c) Design a circuit, including the output signals

159233 Tutorial 4

The following is a block diagram of the picocomputer datapath



Section A

1. What is the purpose of the MAR register?

   a) It holds address of the next instruction.
   b) It holds data to be written to memory
   c) It holds an address for a memory operation
   d) It holds the current instruction

2. Why is the ABR register necessary?

   a) To enable addition and subtraction of two numbers.
   b) To hold the result of an addition or subtraction.
   c) To pass an address to the Accumulator
   d) To invert one of the inputs to the ALU for subtraction.

3. The Picocomputer has an 8 bit data bus and an 8 bit address bus, what would be the approximate maximum size of a typical program (assuming no data storage is necessary)?

   a) 50 instructions b) 80 instructions c) 140 instructions d) 200 instructions

4. What is the sequence of control signals necessary for the ADD instruction (assume opcode has been read already).

   a) OECONST,LDABR→OEPC,LDMAR,OEACC→OEMEM,LDMAR→OEMEM,LDACC
   b) OEACC,LDABR→OEPC,LDMAR,INCPC→OEMEM,LDMAR→OEMEM,LDACC
   c) OECONST,LDABR→OEPC,LDACC,INCPC→OEPC,LDMAR→OEMEM,LDACC
   d) OEACC,LDABR→OEMAR,LDACC,INCPC→OEPC,LDMAR→OEMEM,LDACC

5. What is the sequence of control signals necessary to add one to the accumulator.

   a) OECONST,LDABR→OEPC,LDMAR→OEMEM,LDMAR→OEMEM,LDACC
   b) OECONST,C0/C1,LDABR→OEMEM,LDMAR→OEMEM,LDACC
   c) OECONST,C0/C1,LDABR→OEPC,LDACC→OEPC,LDMAR→OEMEM,LDACC
   d) OECONST,C0/C1,LDABR→OEACC,LDACC

Section B

1. Describe how you would implement the following new pico-computer instruction.

   LDA @address - Load Acc with the value stored in memory at the address given by the value stored in memory at the address specified by the operand (memory indirect addressing).

2. What type of device could be used to construct the Program Counter for the pico-computer?

3. Describe how you would implement the following new pico-computer instruction.

   SWAP *address* - swap the value in the Accumulator with the value in memory at the specified address

4. Programs for the pico-computer will often use self modifying code, briefly explain why? Suggest an extra instruction that could be used to avoid this problem.

159233 Tutorial 5

Section A

1. If the gate delay through a full adder is 2µs and the delay through all other gates is 1us, what is the delay through a 4 bit ripple carry adder?

   a) 4µs b) 5µs c) 8µs d) 12µs

2. What is the delay for question 1 if a 4 bit carry lookahead adder is used?

   a) 4µs b) 5µs c) 8µs d) 12µs

3. A CPU has 300 control signals and a control unit with a microprogram that contains 3000 instructions, what size is the microprogram counter (µPC)?

   a) 8 bits b) 10 bits c) 12 bits d) 14 bits

4. For the CPU in question 3, roughly what size is the microprogram ROM? Assume it uses horizontal microcode.

   a) 942,000 bits
   b) 756,000 bits
   c) 886,000 bits
   d)1024,000 bits

5. A Microprogrammed control unit will generally be slower than a hardwired control unit, why?

   a) Because each instruction will take more steps to execute.
   b) Because the gate delay through the µPC is longer than through the flip flops.
   c) Because the microprogram will be longer than the number of states in the ASM diagram.
   d) Because the gate delay through the microcode ROM and address selection logic is longer than through the multiplexors.

6. What is the next address selection logic in the microcode control unit used for?

   a) Generating the next value for the µPC
   b) Generating the next value for the PC
   c) Reading an instruction from the instruction register.
   d) Holding the address of the current µinstruction in the µcode ROM.

Section B

1. Optimise the picocomputer microcode so that it uses as few microinstructions as possible. Will this speed up the CPU?
2. Briefly explain how you think the microcode patch mechanism in a modern CPU may work.
3. Draw the block diagram for a micocoded control unit that could be used to control the pico-computer.
4. RISC CPUs often contain a hard-wired control unit rather than using microcode, why is this?
5. The picocomputer with a micocoded control unit needed a modified instruction set, why is this?
6. Modify the microcode for the picocomputer to replace the halt instruction with one that increments the accumulator.

Picocomputer microcode:

	μ-addr	cond sel	opcode /addr	next addr	OE CP	LD CP	INC CP	OE CS	ALU CC	LD CC	LD RB	SU BR	LD AM	OE ME	WE ME	DIS AE	
	0	1	.	8	.	.	.	.	.	.	.	.	.	.	.	.	goto fetch
	1	1	.	10	.	.	.	.	.	.	.	.	.	.	.	.	goto LDA
	2	1	.	14	.	.	.	.	.	.	.	.	.	.	.	.	goto ADD
	3	1	.	18	.	.	.	.	.	.	.	.	.	.	.	.	goto SUB
	4	1	.	22	.	.	.	.	.	.	.	.	.	.	.	.	goto STA
	5	1	.	25	.	.	.	.	.	.	.	.	.	.	.	.	goto JPZ
	6	1	.	29	.	.	.	.	.	.	.	.	.	.	.	.	goto CLR
	7	1	.	31	.	.	.	.	.	.	.	.	.	.	.	.	goto HLT
fetch	8	0	.		1	.	.	.	.	.	.	.	1	.	.	.	mar=pc
	9	1	1		.	1	.	.	.	.	.	.	.	1	.	.	pc++; goto Bus
lda	10	0	.		.	.	.	1	.	.	1	.	.	.	.	.	abr=0
	11	0	.		1	.	1	.	.	.	.	.	1	.	.	.	mar=pc; pc++
	12	0	.		.	.	.	.	.	.	.	.	1	1	.	.	mar=m[mar]
	13	1	.	8	.	.	.	.	.	1	.	.	.	1	.	.	a=abr+m[mar]; goto fetch
add	14	0	.		.	.	.	1	.	1	.	.	.	.	.	.	abr=a
	15	0	.		1	.	1	.	.	.	.	.	1	.	.	.	mar=pc; pc++
	16	0	.		.	.	.	.	.	.	.	.	1	1	.	.	mar=m[mar]
	17	1	.	8	.	.	.	.	.	1	.	.	.	1	.	.	a=abr+m[mar]; goto fetch
sub	18	0	.		.	.	.	1	.	1	.	.	.	.	.	.	abr=a
	19	0	.		1	.	1	.	.	.	.	.	1	.	.	.	mar=pc; pc++
	20	0	.		.	.	.	.	.	.	.	.	1	1	.	.	mar=m[mar]
	21	1	.	8	.	.	.	.	.	1	.	1	.	1	.	.	a=abr-m[mar]; goto fetch
sta	22	0	.		1	.	1	.	.	.	.	.	1	.	.	.	mar=pc; pc++
	23	0	.		.	.	.	.	.	.	.	.	1	1	.	.	mar=m[mar]
	24	1	.	8	.	.	.	.	1	.	.	.	.	.	1	.	m[mar]=a;goto fetch
jpz	25	0	.		1	.	1	.	.	.	.	.	1	.	.	.	mar=pc; pc++
	26	2	.	28	.	.	.	.	.	.	.	.	.	.	.	.	if EQZ goto 28
	27	1	.	8	.	.	.	.	.	.	.	.	.	.	.	.	goto fetch
	28	1	.	8	.	1	.	.	.	.	.	.	1	.	.	.	pc=m[mar];goto fetch
clr	29	0	.		.	.	.	1	.	.	1	.	.	.	.	.	abr=0
	30	1	.	8	.	.	.	1	.	1	.	.	.	.	.	.	a=abr+0;goto fetch
hlt	31	1	.	8	.	.	.	.	.	.	.	.	.	.	.	1	disable;goto fetch

159233 Tutorial 6

1. What is an important difference between a five address machine and a three address machine?

   a) The three address machine has a PC, the five address machine does not.
   b) For the three address machine, one source is the same as the destination.
   c) The five address machine has separate jump instructions.
   d) The three address machine has only one type of instruction.

2. Programs for a one address machine will:

   a) Contain more instructions than programs for a two address machine.
   b) Contain fewer instructions than programs for a two address machine.
   c) Use more memory than programs for a two address machine.
   d) Use the same memory as programs for a two address machine.

3. What is the following instruction:

   ADD r1,[r2+r3]

   a) Zero address b) One address c) Two address d) Three address

4. What type of machine is the Pico-computer:

   a) Zero address b) One address c) Two address d) Three address

5. What addressing mode does the second operand in the instruction for question 3 use.

   a) Register b) Register Indirect c) Displacement d) Indexed

6. Which of the following addressing modes does the picocomputer have.

   a) Indexed b) Direct c) Memory Indirect d) Immediate

7. A CPU has instructions using 2 addresses and an ALU operation, the operation is performed on data stored at the two addresses and the result stored at the first address. If an integer a is stored at address 19, what does the following do?

```
XOR 20,20
ADD 20,19
ADD 19,19
ADD 19,20
```

   a) Multiply a by 2 b) Multiply a by 3 c) Multiply a by 4 d) Multiply a by 5

8. Which of the following CPU's will probably run the same program faster

   a) A 3200 Mhz RISC CPU that executes an instruction in 2 clock cycles
   b) A 1600 Mhz CISC CPU that executes an instruction in 1 clock cycle
   c) A 2000 Mhz CISC CPU that executes an instruction in 2 clock cycles
   d) A 2000 Mhz RISC CPU that executes an instruction in 2 clock cycles

159233 Tutorial 7

1. Give a sequence of ARM instructions that will multiply r0 by 20.

2. Write a sequence of ARM instructions to perform the following:

   r0=r0+(r2*1280)+r1*2 (could be useful for A3)

3. What does the following sequence of ARM instructions do?

   ```
   CMP r0,#0
   RSBMI r0,r0,#0
   ```

4. What is the use of the following sequence of ARM instructions?

   ```
   SUB r1,r0,#1
   AND r0,r0,r1
   ```

5. What does the following sequence of ARM instructions do?

   ```
   EOR r0,r0,r1
   EOR r1,r0,r1
   EOR r0,r0,r1
   ```

   (EOR is Exclusive OR)

6. Write a section of ARM assembly language to implement the following C code:

   ```
   while (r0 != r1) {
    if (r0 > r1)
        r0 -= r1;
    else
        r1 -= r0;
   }
   ```

   Try to use as few instructions as possible.

7. Write a complete program in ARM assembly language to find the sum of the first 11 numbers in the Fibonacci sequence (0,1,1,2,3,5,8,13,21,34,55,89). Use registers to hold the current number in the sequence, the previous number and the sum so far. The result should be stored in a memory location labelled “sum”.

Tutorial 8

1. Write ARM assembly language for the following C string functions.

```
// string length
int strlen(char * s) {
        char *sc;

        for (sc = s; *sc != 0; sc++)
            /* do nothing */;
        return sc - s;
}
// concatenate two strings
char * strcat(char * dest, char * src) {
        char *tmp = dest;

        while (*dest != 0)
                dest++;
        while ((*dest++ = *src++) != 0)
            /* do nothing */;

        return tmp;
}

// compare two strings with maximum length
int strncmp(char * s,char *t,int count) {
        char r;

        while (count != 0) {
                r=*s-*t;
                if (r != 0 || *s==0)
                        break;
                s++;
                t++;
                count--;
        }

        return r;
}
/**
 * Find the first substring in a NUL terminated string
 * s1: The string to be searched
 * s2: The string to search for
 */
char * strstr(char * s1,char * s2)
{
        int l1, l2;

        l2 = strlen(s2);
        if (l2==0)
                return s1;
        l1 = strlen(s1);
        while (l1 >= l2) {
                l1--;
                if (!strncmp(s1,s2,l2))
                        return s1;
                s1++;
        }
        return 0;
}
```

Tutorial 8 Solutions

```
@ solutions for 159.233 Tutorial 8
@ Martin Johnson 2010

.global _start
@ find the length of the string passed in r0
@ returns the length in r0
strlen:         stmfd sp!,{r1}          @ save r1
                mov r1,r0               @ r1 is a pointer to the start of the string
lenloop:        ldrb r12,[r1],#1        @ get next char and increment r1
                cmp r12,#0              @ is it the NUL char
                bne lenloop             @ if not then keep going
                sub r0,r1,r0            @ calculate length
                sub r0,r0,#1            @ adjust for NUL char
                ldmfd sp!,{r1}          @ restore r1
                mov pc,lr               @ return

@ concatenate two strings
@ r0 is destination, r1 is source
@ r0 and r1 are unchanged
strcat:         stmfd sp!,{r0,r1}       @ save r0 and r1
                                        @ find end of dest string
catloop1:       ldrb r12,[r0],#1        @ get next char and incremet dest
                cmp r12,#0              @ is it the NUL char
                bne catloop1            @ if not then keep going
                sub r0,r0,#1            @ adjust for NUL char
catloop2:       ldrb r12,[r1],#1        @ copy char at src to
                strb r12,[r0],#1        @ dest and increment src and dest
                cmp r12,#0              @ was is the NUL char
                bne catloop2            @ if not, carry on
                ldmfd sp!,{r0,r1}       @ restore r0 and r1
                mov pc,lr               @ return

@ compare 2 strings with maximum length
@ string pointers are in r0 and r1, r2 is the length
strncmp:        stmfd sp!,{r1-r3}       @ save registers
                b cmptest               @ while loop
cmploop:        ldrb r12,[r0],#1        @ get next chars
                ldrb r3,[r1],#1
                subs r3,r12,r3          @ compare them
                bne stopcmp             @ stop if not the same
                cmp r12,#0              @ chack for NUL char
                beq stopcmp
                sub r2,r2,#1            @ decrement length
cmptest:        cmp r2,#0               @ repeat until 0
                bne cmploop
stopcmp:        mov r0,r3               @ return difference
                ldmfd sp!,{r1-r3}       @ restore registers
                mov pc,lr               @ return

@ find the first substring in a NUL terminated string
@ r0 points to the string to be searched (s1)
@ r1 points to the string to search for (s2)
@ return a pointer to the found string
strstr:         stmfd sp!,{r1-r5,lr}    @ save registers
                mov r4,r0               @ r4 is s1
                mov r5,r1               @ r5 is s2
                mov r0,r5
                bl strlen               @ get length of s2
                mov r2,r0               @ r2 is l2
                cmp r0,#0               @ if l2 is 0
                mov r0,r4               @ set return value to s1
                beq leavestrstr         @ return
                bl strlen               @ get length of s1
                mov r3,r0               @ r3 is l1
                b strtest               @ while loop
strloop:        sub r3,#1               @ decrement l1
                mov r0,r4               @ call strncmp(s1,s2,l2)
                mov r1,r5
                bl strncmp
                cmp r0,#0               @ check if found
                moveq r0,r4             @ if so, set return value to s1
                beq leavestrstr         @ and return
                add r4,r4,#1            @ increment s1
strtest:        cmp r3,r2               @ check l1>=l2
                bge strloop             @ keep going while it is
                mov r0,#0               @ set return value to 0
leavestrstr:    ldmfd sp!,{r1-r5,pc}    @ return
```

```
@ print a newline char on the serial port
newline:        stmfd sp!,{r0,lr}
                adr r0,cr
                bl printstr
                ldmfd sp!,{r0,pc}

_start:         bl hardware_init        @ initialise the hardware
                adr r0,string           @ get strlen(string)
                bl strlen               @ test strlen function
                bl printhex8            @ print length of string
                bl newline
                adr r0,string           @ do strcat(string,world)
                adr r1,world
                bl strcat               @ test strcat function
                bl printstr
                bl newline
                adr r1,more             @ do strcat(string,more)
                bl strcat               @ try another strcat
                bl printstr
                bl newline
                adr r0,string           @ do strncmp(string,world,16)
                adr r1,world
                mov r2,#16
                bl strncmp              @ test strncmp function
                bl printhex8
                bl newline
                adr r0,same             @ do strncmp(same,world,16)
                bl strncmp              @ test another strncmp
                bl printhex8
                bl newline
                adr r0,string           @ do strstr(string,name)
                adr r1,same
                bl strstr               @ test strstr function
                bl printstr
end:            b end


string:         .asciz "Hello "
.space 64
world:          .asciz "World"
cr:             .asciz "\n\r"
same:           .asciz "World"
more:           .asciz " of ARM assembly"
```