### Fall 2021 Courses – Bryn Mawr College

- CMSC 110 Introduction to Computing (TBA)
- CMSC 113 Computer Science 1 (Chris Murphy)
- CMSC 113 Computer Science 1 (Chris Murphy)
- CMSC 206 Data Structures (Geoff Towell)
- CMSC 206 Data Structures (TBA)
- CMSC 231 Discrete Mathematics (Aline Normoyle)
- **CMSC 245 Principles of Programming Languages (Geoff Towell)**
- CMSC 317 Computer Animation (Aline Normoyle)
- CMSC 340 Analysis of Algorithms (Dianna Xu)
- CMSC 355 Operating Systems (Dianna Xu)
- CMSC 372 Artificial Intelligence (Deepak Kumar)

### Spring 2022 Courses – Bryn Mawr College [DRAFT]

- CMSC 113 Computer Science 1 (Deepak Kumar)
- CMSC 206 Data Structures (Geoffrey Towell)
- CMSC 231 Discrete Mathematics (TBA)
- **CMSC 246 Systems Programming (Chris Murphy)**
- CMSC 240 Principles of Computer Organization (Aline Normoyle)
- **CMSC 283 Topics in Computer Science (TBA)**
- CMSC 330 Algorithms: Design & Practice (Deepak Kumar) ✓
- CMSC 380 Recent Advances in Computer Science
- **Topic: Database Systems in practice (Geoffrey Towell)**
- CMSC 353 Software Engineering (Chris Murphy)
- CMSC 399 Senior Conference (Aline Normoyle)

### FALL 2021 - Haverford College

- CMSC 105 Introduction to CS
- CMSC 231 Discrete Mathematics
- **CMSC 245 Prin. Of Programming Languages**
- CMSC 260 Intro. To Data Science (??)
- CMSC 325 Computational Linguistics
- CMSC 399 Senior Conference

### NOTE:

We are also in the process of changing some course numbers.
The courses will be the same but different numbers.
Program to add 12 numbers stored in x3100..x310B
Result will be in R3

![](image)

```
0011 0000 0000 0000     ; start program at x3000
1110 001 011 111 111    ; R1 = x3100
0101 011 011 1 00000    ; R3 = 0
    ; R2 = 12
0101 010 010 1 00000    ; R2 = 0
0001 010 010 1 01100    ; R2 = R2 + 12
* if R2 = 0 Go to **
0000 010 00000101      ; Branch if Zero
0110 100 001 00000     ; R4 = M[R1 + 0]
0001 011 011 0 00 100    ; R3 = R3 + R4
0001 001 011 1 00001    ; R1 = R1 + 1
0001 010 010 1 11111    ; R2 = R2 - 1
0000 111 111111010      ; Go to * [PC + #6]
1111 0000 001 00101     ; ** HALT
```

### Assembly Language

1. Each opcode has a name: ADD, AND, LD, LDR, etc.
2. Each operand has a name: R0, R1, ..., R7
   Or, is a literal: #22 (decimal), x16 (hexadecimal), b10110 (binary)
   Or, is a location:
   How to code: R1 = x3100
   R4 = M[R1 + 0]
   Use a label!
3. Each instruction can have a label, if needed.
   E.g. LEA R1, DATA
   LDR R4, R1, #0
4. Comments: same as in machine language programs.

```
.START
LEA R1, DATA
ADD R2, R2, #12
LDR R4, R1, #0
ADD R3, R3, R4
ADD R1, R1, #1
ADD R2, R2, #1
BR DONE
DONE
HALT

HALT = DONE
```

; Program to add 12 numbers stored in x3100..x310B
; Result will be in R3
0011 0000 0000 0000     ; start program at x3000
1110 001 011 111 111    ; R1 = x3100
0101 011 011 1 00000    ; R3 = 0
    ; R2 = 12
0101 010 010 1 00000    ; R2 = 0
0001 010 010 1 01100    ; R2 = R2 + 12
* if R2 = 0 Go to **
0000 010 00000101      ; Branch if Zero
0110 100 001 00000     ; R4 = M[R1 + 0]
0001 001 011 1 00001    ; R1 = R1 + 1
0001 010 010 1 11111    ; R2 = R2 - 1
0000 111 111111010      ; Go to * [PC + #6]
1111 0000 001 00101     ; ** HALT = DONE
0000 111 111111111111111111
```

Each opcode has a name: ADD, AND, LD, LDR, etc.
Each operand has a name: R0, R1, ..., R7
Or, is a literal: #22 (decimal), x16 (hexadecimal), b10110 (binary)
Or, is a location:
How to code: R1 = x3100
R4 = M[R1 + 0]
Use a label!
Each instruction can have a label, if needed.
E.g. LEA R1, DATA
LDR R4, R1, #0
Comments: same as in machine language programs.
The complete Program

; Program to add 12 numbers stored in x3100..x310B
; Result will be in R3
; start program at x3000

.ORIG x3000

START
LEA R1, DATA ; R1 = x3100
AND R3, R3, #0 ; R3 = 0
AND R2, R2, #0 ; R2 = 0
ADD R2, R2, #12 ; R2 = R2 + 12

; * if R2 = 0 Go to **
LOOP
BRz DONE ; Branch if Zero
LDR R4, R1, #0 ; R4 = M[R1 + 0]
ADD R3, R3, R4 ; R3 = R3 + R4
ADD R1, R1, #1 ; R1 = R1 + 1
ADD R2, R2, #-1 ; R2 = R2 - 1
BR LOOP ; Go to * [PC + # -6]

DONE
HALT ; ** HALT

.END

; The data for the program starts at x3100

.ORIG x3100

DATA
.FILL 1
.FILL 2
.FILL 3
.FILL 4
.FILL 5
.FILL 6
.FILL 7
.FILL 8
.FILL 9
.FILL 10
.FILL 11
.FILL 12
.END

.data
BLKW 10
A .BLKW 1
B .BLKW 1
SOM .BLKW 1

.string$s
HELLO C526
.MESG
"H"
"e"
"l"
"o"
"n"
null-terminated strings
Problem: Compute the sum

\[ R_3 = n_1 + n_2 + n_3 + \ldots \]

We do not know how many numbers are there. They are stored starting from x3100. But a sentinel value of -1 will indicate the end.

Write the program starting from location x3000.

```
.ORIG x3000
START
  LEA R1, DATA
  AND R3, R3, #0
  LDR R4, R1, #0
  LOOP
    BRN STOP
    ADD R3, R3, R4
    ADD R1, R1, #1
    LDR R4, R1, #0
    BR LOOP
  STOP
  HALT
.END

DATA
.ORIG x3100
  .FILL —
  .FILL —
  .FILL —
  .FILL —
  .FILL -1
.END
```
; Program to add 12 numbers stored in x3100..x310B
; Result will be in R3
; start program at x3000
.ORIG    x3000

START
  LEA    R1, DATA          ; R1 = x3100
  AND    R3, R3, #0        ; R3 = 0
        ; R2 = 12
  AND    R2, R2, #0        ; R2 = 0
  ADD    R2, R2, #12       ; R2 = R2 + 12

; * if R2 = 0 Go to **
LOOP
  BRz    DONE              ; Branch if Zero
  LDR    R4, R1, #0        ; R4 = M[R1 + 0]
  ADD    R3, R3, R4        ; R3 = R3 + R4
  ADD    R1, R1, #1        ; R1 = R1 + 1
  ADD    R2, R2, #1        ; R2 = R2 - 1
  BRmp   LOOP              ; Go to * [PC + #6]
DONE
  HALT              ; ** HALT

.END

0011 0000 0000 0000
1110 001 011 111 111
0101 011 011 1 00000
0101 010 010 1 00000
0001 010 010 1 01100

Second Pass

addTenIns.asm

START:  x3000
DATA:   ?? x3100
LOOP:   x3004
DONE:   ?? x300A

Symbol Table
R0:  00D
R1:  001
R2:  161
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Action</th>
<th>Addressing Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD R2, R2, R3</td>
<td>R2 = R2 + R3</td>
<td>Register</td>
</tr>
<tr>
<td>ADD R2, R2, #1</td>
<td>R2 = R2 + 1</td>
<td>Immediate</td>
</tr>
<tr>
<td>AND R2, R2, R3</td>
<td>R2 = R2 AND R3</td>
<td>Register</td>
</tr>
<tr>
<td>AND R2, R2, #0</td>
<td>R2 = R2 AND 0</td>
<td>Immediate</td>
</tr>
<tr>
<td>BR[n][z][p] LABEL</td>
<td>If [n][z][p] Go to LABEL</td>
<td>CC, PC-relative</td>
</tr>
<tr>
<td>JMP R1</td>
<td>PC = R1</td>
<td>Register</td>
</tr>
<tr>
<td>JSR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JSRR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LD R2, LABEL</td>
<td>R2 = m[LABEL]</td>
<td>Register, PC-relative</td>
</tr>
<tr>
<td>LDI R2, LABEL</td>
<td>R2 = m[m[LABEL]]</td>
<td>Register, Indirect</td>
</tr>
<tr>
<td>LDR R2, R0, #n</td>
<td>R2 = m[R0+n]</td>
<td>Base Register</td>
</tr>
<tr>
<td>LEA R2, LABEL</td>
<td>R2 = LABEL</td>
<td>Register, PC-relative</td>
</tr>
<tr>
<td>NOT R2, R1</td>
<td>R2 = NOT(R1)</td>
<td>Register</td>
</tr>
<tr>
<td>RET</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ST R2, LABEL</td>
<td>m[LABEL] = R2</td>
<td>Register, PC-relative</td>
</tr>
<tr>
<td>STI R2, LABEL</td>
<td>m[m[LABEL]] = R2</td>
<td>Register, Indirect</td>
</tr>
<tr>
<td>STR R2, R0, #n</td>
<td>m[R0 + n] = R2</td>
<td>Register, Base Register</td>
</tr>
<tr>
<td>HALT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

### Assembly Cheat Sheet

- **Friday, April 16, 2021**
- **9:30 AM**
- **240 Lectures Page 7**