Goals

1 **Simulate the operation of a Direct Mapped Cache Controller.**

2 **The cache controller can be configured in any 1 of 12 different modes.**

3 **The controller will simulate the handling of the cache mapping for a bubble sort algorithm in each of the 12 different controller modes.**
(The data is given in a memory trace file)

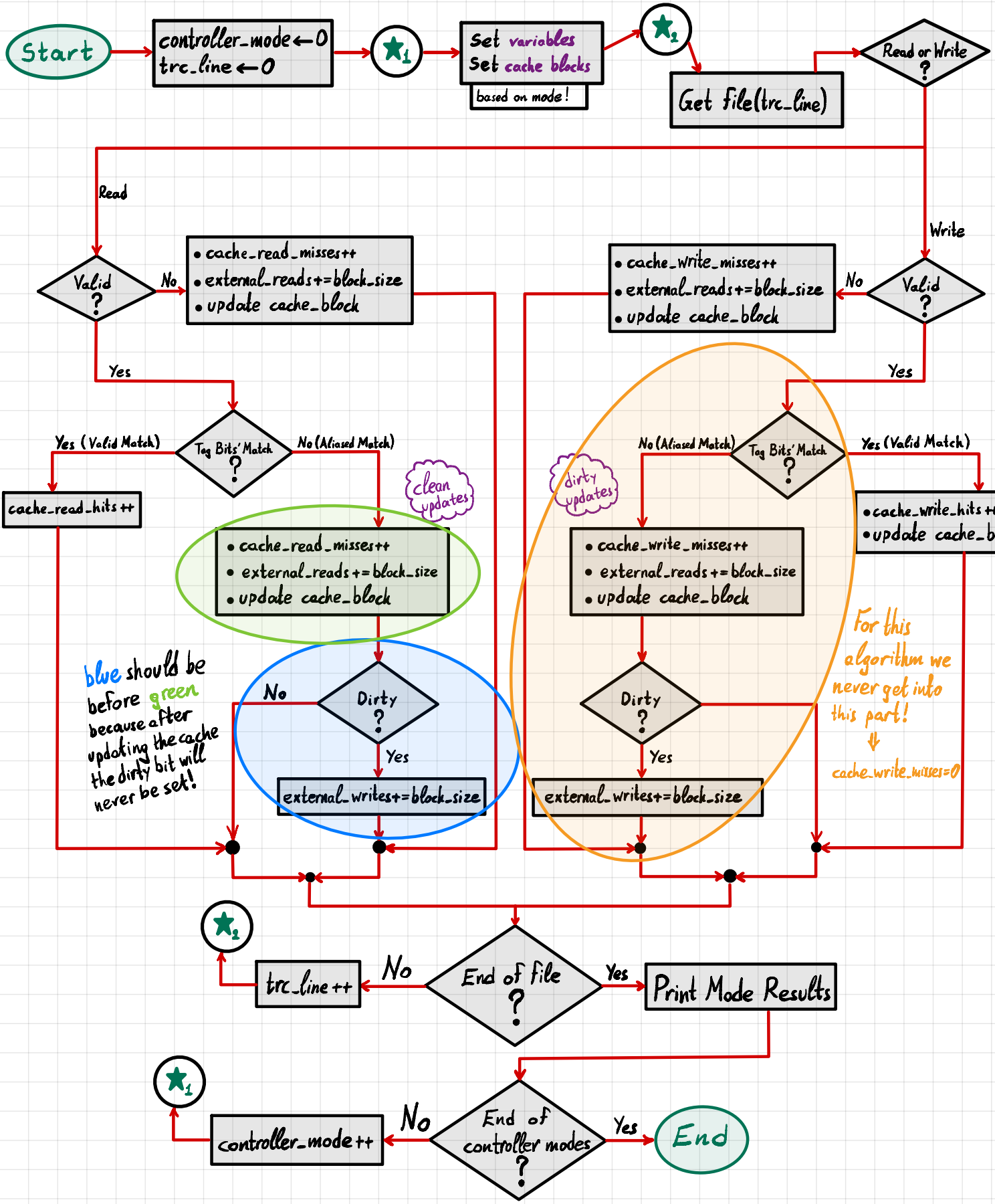
4 **Cache controller in a single source file in C.**
It should be able to be run from the terminal window.

5 **Produce the simulation results from my own memory trace file (#74)**
(.csv file in ASCII format)

6 **Produce a report with:**

- a **Result Presentation**
  - a.1 **Graphical Format**
  - a.2 **Tabular Format**
- b **Result Analysis**
  Comment on performance based on various metrics
  (read/write/total hit/miss ratio, access efficiency, overall speedup)
  Remember to refer to the pattern of memory accesses when executing the bubble sort algorithms on the embedded processor.

Direct Mapped Controller Flowchart



Print Mode Results

trace_file_name	bubble_sort_trace_74.trc (same for all controller modes)
mode_ID	obvious
NRA	external_reads
NWA	external_writes
NCRH	cache_read_hits
NCRM	cache_read_misses
NCWH	cache_write_hits
NCWM	cache_write_misses

cache-block objects:
tag_bits, dirty_bit, valid_bit
structs or 3 arrays or 1 array

variables: external_reads/writes
cache_read_hits/misses cache_write_hits/misses
controller_mode, trc_line
block_size, cache_size, num_of_cache_blocks

updating seems synonymous to evicting.

To comment on the performance the simulation gather data along the way:
- total memory accesses
- total cache hits
- total cache misses
- total read accesses
- total write accesses

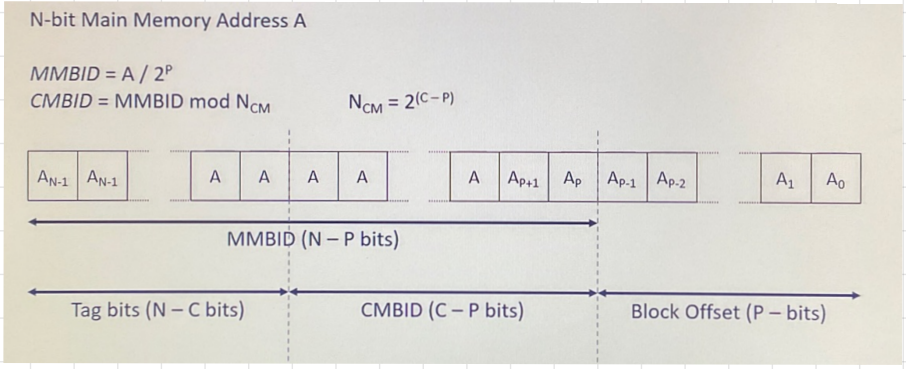
mma → main memory address

From this we can get

MMBID = (mma) div (# of addresses / block) → block_size

Tag Bits = (MMBID) div (# of blocks in cache) → num_of_cache_blocks

CMBID = (MMBID) mod (# of blocks in cache)



Metrics

$$\text{Cache Hit Ratio} = \frac{\text{cache accesses}}{\text{total accesses}} \longrightarrow \text{cache\_read\_hits} + \text{cache\_write\_hits}$$

$$\text{Cache Read Hit Ratio} = \frac{\text{cache read hits}}{\text{total reads}}$$

$$\text{Cache Write Hit Ratio} = \frac{\text{cache write hits}}{\text{total writes}}$$

$$\text{Cache Miss Ratio} = 1 - \text{Cache Hit Ratio}$$

$$\text{Cache Read Miss Ratio} = 1 - \text{Cache Read Hit Ratio}$$

$$\text{Cache Write Miss Ratio} = 1 - \text{Cache Write Hit Ratio}$$

Access Efficiency → $e = \frac{T_{CM}}{T_{AV}}$ (Amdahl's Law) $e = \frac{1}{1 + (1-h) \times S}$, $S = \frac{T_{MM}}{T_{CM}}$

T_{CM}: cache access time. T_{AV}: average access time.

$$\text{Overall Speedup} = \frac{1}{(1-h) + h/S}, \quad S = \frac{T_{MM}}{T_{CM}}$$

For the purpose of calculating execution times you should assume that the main memory has access time of 80 ns, and the cache memory has an access time of 8 ns.

↳ Use this & the data to calculate total execution time (ultimate metric)

+ Comment on assumptions! (e.g. No early restart OR critical word first)

⇓

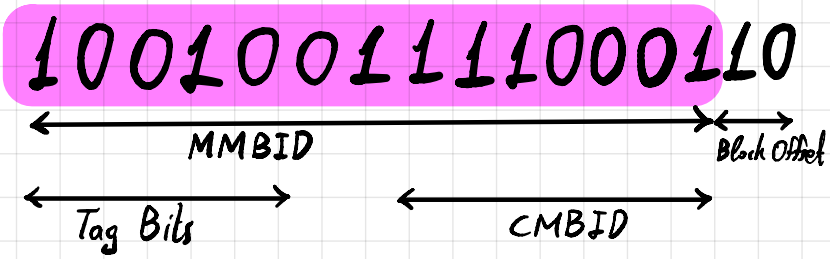
The time we get will be the worst case scenario.

which of the metrics best describes execution time?

For example: mma = $1001001111000110_{bin} = 37830_{dec}$

	block_size	num_of_cache_blocks	MMBID	Tag Bits	CMBID
Mode 1	4	128	9457	73	113
Mode 2	8	64			
Mode 3	16	32			
Mode 4	32	16			
Mode 5	64	8			
Mode 6	128	4	256		
Mode 10	16	1024	1024	1	0

So



address / smth

(div) gives MSBs

(mod) gives LSBs

↘ address % smth

block_size = 2^p

⇓

Block Offset = p - bits

⇓

MMBID = (16-p) bits

num_of_cache_blocks = 2^c

⇓

CMBID = c-bits

⇓

Tag Bit = (16-p-c) bits