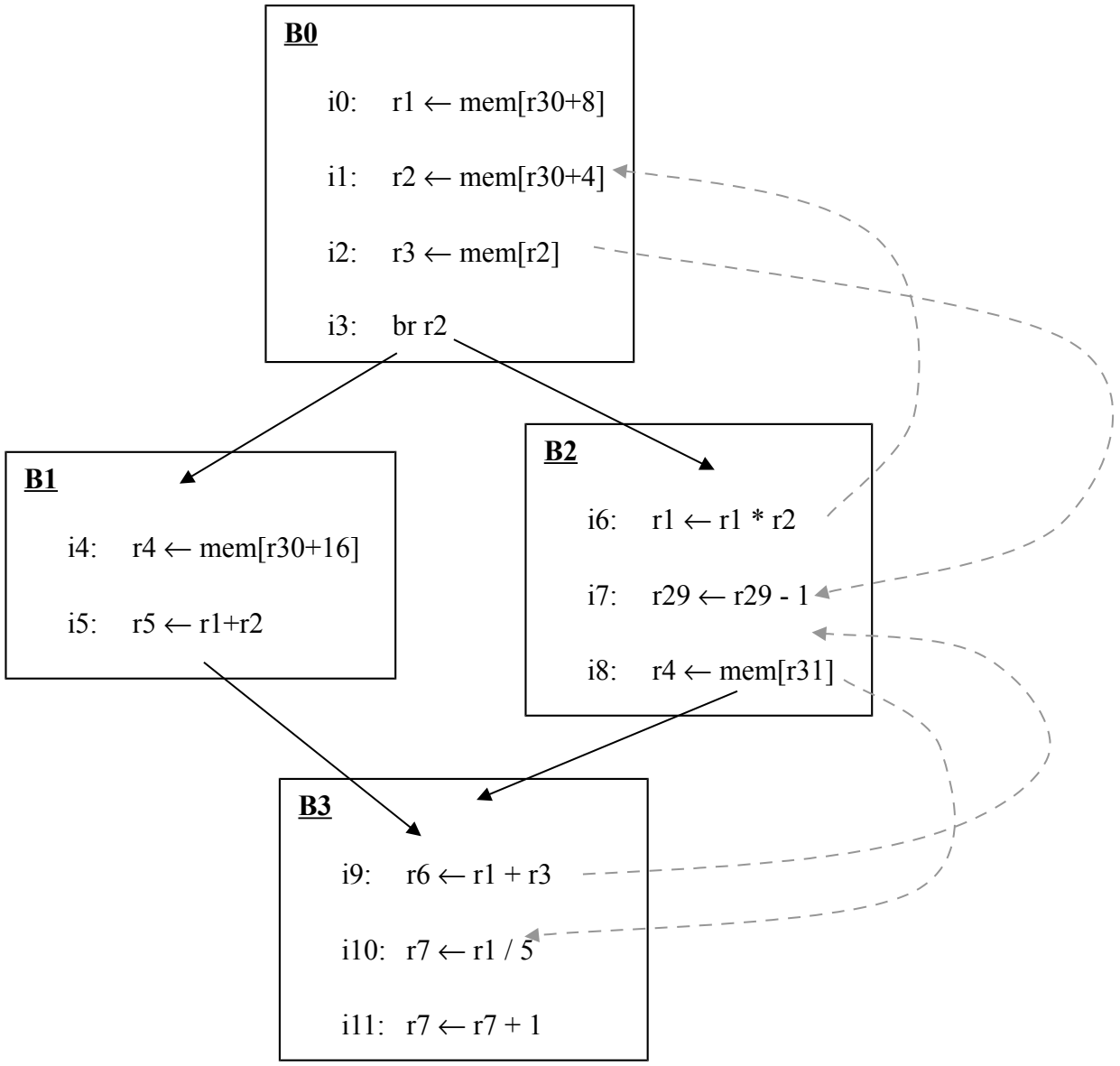
Homework 3
18640- Fall 2014
Due Date – 11/26/2014

1. Code Motion **20 points**

Consider the following control-flow graph with 4 nodes connected by fork and join edges.



We want to reschedule the instructions under the assumption that B0 is followed by B2 more than 95% of the time. Our scheduler has indicated that the following code motions are required:

1. i2 should be moved to before i7 in B2
2. i6 should be moved to between i1 and i3 in B0
3. i8 should be moved to before i10 in B3
4. i9 should be moved to after i7 in B2

The code motions are shown as dashed edges in the control-flow graph above. Following code motion, the most common execution path **must** correspond to i0, i1, i6, i3, i2, i7, i9, i8, i10, i11.

Modify and complete the control-flow graph to reflect the transformed code after the code motions. Your transformations must preserve the original functionality of the program segment *exactly.* You can assume this control flow graph can only be entered from the beginning of B0. You can assume registers r10 ~ r19 are free temporary registers, but all other registers are assumed to be alive before B0 and after B3. In the transformed control-flow graph, you are allowed to enter a node from the beginning and in the middle.

2. Power Modeling 20 points

Assume a processor with the following energy consumption characteristics for the event types listed.

Event	Energy per instruction (nJ)
Fetch, Decode, Dispatch	**2.0**
Issue	**1.3**
Reg file read (per instruction)	**1.1**
Int/branch execute	**0.4**
Load execute	**0.9**
Store execute	**0.6**
Reg file write	**0.65**
Commit	**0.3**

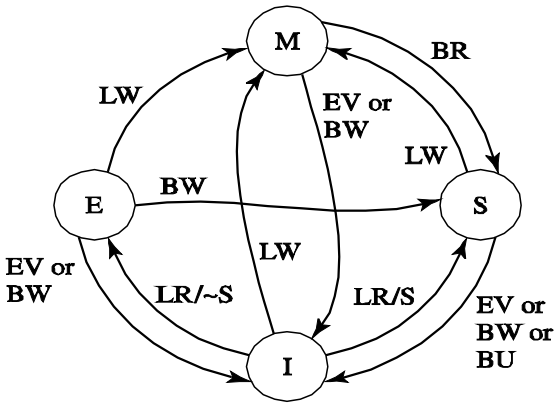
Assume an instruction mix of 25% loads, 15% stores, 20% branches and jumps, and 40% integer ALU instructions. Assume that all loads and integer ALU ops write a register, and that this is a physical-register file machine where registers are written speculatively right after an instruction executes. Finally, assume the processor runs at 2GHz.

a. Assume that on average the processor commits 1.4 IPC, executes 1.8 IPC, issues 2.0 IPC, and fetches/decodes/dispatches 3.0 IPC. What is the power consumption (in watts) of this processor running this workload? 8 points

b. You decide to implement a branch confidence estimator that slows down fetch whenever you have predicted a branch with low confidence. Assume that the power consumption of the confidence estimator is negligible. This approach reduces the commit rate to 1.35 IPC, execution rate to 1.7 IPC, issue rate to 1.8 IPC, and fetch/decode/dispatch rate to 2.75 IPC. Now what is the power consumption? 8 points

c. Given your answer for (b), does your confidence scheme pass or fail the 3:1 rule of thumb?
(**3:1 rule of thumb:** +/- 1% performance buys -/+ 3% power – Eg: If new technique saves 7% power and performance degrades by 2%. It is useful because 7 > 3 * 2, If new technique saves 10 % power and if the performance degrades by 5%, it is not useful since 10 < 3 * 5) 4 points

3. Modern Cache Coherence protocols 20 points



Many modern systems use a MOESI cache coherence protocol, where the semantics of the additional O state are that the line is shared-dirty: i.e., multiple copies may exist, but the other copies are in S state, and the cache that has the line in O state is responsible for writing the line back if it is evicted.

Current State s	**Event and Local Coherence Controller Responses and Actions (s' refers to next state)**					
	Local Read (LR)	**Local Write (LW)**	**Local Eviction (EV)**	**Bus Read (BR)**	**Bus Write (BW)**	**Bus Upgrade (BU)**
Invalid (I)	Issue bus read if no sharers then s' = E else s' = S	Issue bus write s' = M	s' = I	Do nothing	Do nothing	Do nothing
Shared (S)	Do nothing	Issue bus upgrade s' = M	s' = I	Respond shared	s' = I	s' = I
Exclusive (E)	Do nothing	s' = M	s' = I	Respond shared s' = S	s' = I	Error
Modified (M)	Do nothing	Do nothing	Write data back; s' = I	Respond dirty; Write data back; s' = S	Respond dirty; Write data back; s' = I	Error

a. Explain what benefit accrues from the addition of O state to the MESI protocol. 4 points

b. **Redraw** the state diagram and fill the table below to accommodate the O state.

12 points

Current State s	**Event and Local Coherence Controller Responses and Actions (s' refers to next state)**					
	Local Read (LR)	**Local Write (LW)**	**Local Eviction (EV)**	**Bus Read (BR)**	**Bus Write (BW)**	**Bus Upgrade (BU)**
Invalid (I)		Issue bus write s' = M	s' = I	Do nothing	Do nothing	Do nothing
Shared (S)	Do nothing	Issue bus upgrade s' = M	s' = I	Respond shared	s' = I	s' = I
Exclusive (E)	Do nothing	s' = M	s' = I	Respond shared s' = S	s' = I	Error
Owned (O)						
Modified (M)	Do nothing	Do nothing	Write data back; s' = I			Error

c. The base protocol has two snoop responses (shared and dirty). Does the addition of the 'O' state require any new responses? If so, what are they? 4 points

4. Cache Coherence 20 points

Consider a shared-memory multiprocessor that consists of three processor/cache units and where cache coherence is maintained by an MSI protocol. This table shows the access sequence taken by three processors to the same block but to different variable A, B, and C in that block.

	Processor 1	**Processor 2**	**Processor 3**
1	RA		
2		RB	
3			RC
4	WA		
5			RC
6		RA	
7	WB		
8			RA
9			RB

a**.** Classify misses with respect to cold, true sharing, and false sharing misses. (We have done the first one for you).

	Processor 1	**Processor 2**	**Processor 3**	**Miss type**
1	RA			Cold Miss
2		RB		
3			RC	
4	WA			
5			RC	
6		RA		
7	WB			
8			RA	
9			RB	

b. Which of the misses could be ignored and still guarantee that the execution is correct?

c. Determine the fraction of essential traffic resulting from the access sequence using the parameters from the previous table, and assuming that the block size is 32 bytes.

5. Short answers (4 points each) 20 points

a. Trace scheduling vs Trace Caching

b. Power Optimization vs Energy Optimization

c. Shared memory model vs Message passing model

d. Multiprocessor Architecture vs Multicomputer Architecture

e. Code locking vs Data locking